Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE)

Trabajo de Diploma

Título: *"Componente para la réplica de ficheros en el Replicador de datos REKO"*

Autor: Walter Felix Jacas Jardines

Tutor: Ing. Frank Rosales Muñoz

Cotutor: Dr.C. Jorge Sergio Menéndez Pérez

La Habana, Junio de 2015

Año 57 de la Revolución

DECLARACIÓN JURADA

Declaro ser el autor del presente trabajo de diploma y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo a los ____ días del mes de _____ del año 2015.

Firma del autor:

_____

Walter Felix Jacas Jardines.

Firma de los tutores:

_____ _____

Ing. Frank Rosales Muñoz. Dr.C. Jorge Sergio Menéndez Pérez.

DATOS DE CONTACTO

Walter F. Jacas Jardines.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Correo: wfjacas@estudiantes.uci.cu

Tutores
Ing. Frank Rosales Muñoz.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Correo: frankr@uci.cu

Dr.C. Jorge Sergio Menendéz Pérez.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Correo: smenendez@uci.cu

AGRADECIMIENTOS

Para lograr que este sueño sea realidad hay varias personas que de una forma u otra han aportado su granito de arena en estos 5 años de universidad, a todos de corazón le estoy muy agradecido por el apoyo que me brindaron. Ahora listo a varias personas que quiero y aprecio, por favor el que se me quede, debe estar claro de mi eterna gratitud por estos 5 años.

A mi madre que es mi razón de ser, mi ejemplo y mi guía seguir en lo profesional y personal, por todo el apoyo brindado desde que nací.

A mi padre por todo el apoyo que me ha brindado desde que nací, por ser un ejemplo a seguir.

A mi tía la Gordi, Yoryi, gracias por tu cariño y apoyo brindado desde que nací.

A mi tío Nestor, Jardi gracias por siempre estar ahí disponible, gracias por tu cariño y haberme apoyo estos largos 5 años de la universidad.

A mi tío Víctor, gracias por tus pláticas, por tus consejos y ayuda incondicional.

A mi tía Lucy, gracias por tu cariño, por tu preocupación, y por tu apoyo incondicional en toda la universidad.

A mi tía Martha y Arturo por su apoyo y cariño durante toda mi etapa estudiantil.

A Miriam, Mey, Teresita y familiares de cada uno, por haberme aceptado como un miembro más de su familia en estos 5 años, por sus consejos, cariño y ayuda incondicional.

A Dany, Javi, Carmen, Ramón y familia, por su cariño, apoyo, y ser parte de esta gran familia de los Jardines.

A Sayas, por ser un ejemplo digno a seguir, por sus consejos, cariño y apoyo incondicional.

A mis primos, a Laura, la Temba, en fin a toda mi familia, por su apoyo y cariño en toda esta etapa.

A mi vecina Cari por escucharme, brindarme sus sabios consejos y facilitarme su amor de madre.

A Yumi por haberme soportado en todo este período estresante de tesis, por su cariño y amor incondicional.

A la profe Siomara y Yeni su hija por todo el cariño y apoyo brindado desde el primer año de la universidad.

A la profe Zaida por todo el cariño y ayuda brindada desde que comenzó a ser profe guía.

A mis compañeros de carrera, por todas las horas de estudio y conocimientos que hemos compartido.

A los colegas de primer año, a los del apto y de Santiago, como son David, Rioger, Cuba, Coco, Bravo, Alexis, en fin a todos los que desde primer año me han brindado su amistad incondicional.

A todos los profesores que me educaron y ayudaron en estos cinco años y en toda mi vida estudiantil. Les agradezco la paciencia con la que me educaron.

A mis tutores por su paciencia y consejos brindados en todo el período de tesis, al equipo de Reko y a todos los profesores que me ayudaron en los cortes y pre defensas de este trabajo de diploma.

A los miembros del tribunal por la ayuda brindada en el período de tesis.

A todas mis amistades de santiago.

A Fidel, eterno líder de esta inmensa revolución, por hacer realidad el proyecto UCI.

A todos muchas GRACIAS.

DEDICATORIA

A mi madre, por ser mi razón de ser diaria, gracias por siempre estar a mi lado, gracias por brindarme tus sabios consejos cuando más lo necesito, gracias mamita por brindarme tu amor incondicional, gracias por brindarme tu hombro cuando más lo necesito, gracias por ayudarme a encontrar siempre el camino correcto y ayudarme a salir del bache, gracias a ti este sueño se ha hecho realidad, cada meta, cada gloria lograda te lo debo a ti. Gracias mi puchunga por existir 100% para mí. Gracias por ser mi guía y mi ejemplo tanto en el campo profesional como personal, gracias por formarme con valores y conceptos para la vida.

A mi padre por su apoyo incondicional, en la vida, por sus consejos y por formarme un hombre con valores en la vida. Por sus consejos en el momento que más lo necesito, por ayudarme a encontrar el camino correcto, y siempre darme ánimos para seguir adelante cuando más lo necesito.

A la memoria de mis abuelos, que siempre de pequeño me brindaron su amor y cariño incondicional, y su mayor sueño era verme graduado en la universidad y a mi primo Alfrank graduado de médico.

RESUMEN

La tendencia al desarrollo de sistemas distribuidos ha aumentado la necesidad de la utilización de sistemas de réplicas. Estos últimos, permiten mantener actualizados los objetos de las Bases de Datos, lo que permite la recuperación además de la tolerancia a fallos en la distribución de los datos. En la Universidad de las Ciencias Informáticas, se desarrolló en el año 2007, el Replicador de datos REKO, proponiendo una herramienta multiplataforma capaz de cubrir las necesidades de replicación, dígase sincronización, centralización, protección, así como la recuperación de la información. En el año 2010 se le añade a REKO un módulo para la réplica de fichero, el cual no cuenta con un mecanismo de respaldo en caso de fallo del Servicio de Protocolo de Transferencia de Ficheros (FTP, por sus siglas en inglés). Sólo responde a una topología de red en estrella, y para su funcionamiento, se hace necesario el uso de un servidor FTP.

El objetivo que se persigue con el presente trabajo es desarrollar un componente que permita la replicación de ficheros. La solución propuesta garantiza la fiabilidad, calidad y consistencia de los datos en un escenario de réplicas para el software REKO. Para la implementación de la solución se empleó como lenguaje de programación Java, el entorno de desarrollo usado fue Eclipse STS, y como metodología de desarrollo el Proceso Unificado Ágil, con UML como lenguaje de modelado. La solución implementada provee el servicio de réplica de ficheros para los gestores de bases de datos PostgreSQL, MySQL, Oracle y SQLServer.

Palabras clave: base de datos distribuida, réplica de fichero.

ÍNDICE DE CONTENIDOS

ÍNIDICE DE TABLAS

ÍNIDICE DE FIGURAS

INTRODUCCIÓN

Los sistemas informáticos o sistemas automatizados de información existen desde que comenzaron a utilizarse las computadoras en la gestión económica, además de emplearse en otras esferas de la vida humana. En cuanto a su arquitectura pueden ser agrupados en tres clasificaciones; la primera de ellas, constituidas por los Sistemas Informáticos Centralizados; la segunda, responde a los Sistemas Informáticos Descentralizados; y por último, los Sistemas Informáticos Distribuidos, compuestos por un "*conjunto de equipos de tamaños y características variables que pueden estar perfectamente conectados entre sí"*. Algunos de estos sistemas plantean en su arquitectura el uso de varios servidores de bases de datos separados geográficamente, que deben estar actualizados y sincronizados entre sí, lo cual puede lograrse mediante la réplica de datos. (1)

La tendencia al desarrollo de sistemas distribuidos ha aumentado la necesidad de la utilización de sistemas de réplicas. Estos últimos, permiten mantener actualizados los objetos de las Bases de Datos (BD); lo que permite la recuperación y la tolerancia a fallos en la distribución de los datos.

La Universidad de las Ciencias Informáticas (UCI), con el objetivo de llevar a cabo el proceso de réplica de datos sobre la distribución de Oracle 10g en el Sistema de Gestión Penitenciario Venezolano (SIGEP), desarrolló en el año 2007, el Replicador de datos REKO proponiendo una herramienta multiplataforma capaz de cubrir las necesidades de replicación, dígase sincronización, centralización, protección y recuperación de la información.

En el año 2010 se decide enriquecer el software, añadiéndole funcionalidades para la replicación de ficheros, tales como imágenes y videos. Ello era necesario para su uso en el Sistema Informático para la gestión de información de las Coordinaciones Regionales, del proyecto Prevención del Delito, perteneciente al convenio Cuba-Venezuela IX Mixta, añadiéndole a REKO un módulo para el manejo de ficheros, que permite replicar archivos de gran tamaño en un escenario de réplica. Dicho módulo requiere un servicio de Protocolo de Transferencia de Archivos (FTP, por sus siglas en inglés) para replicar ficheros que exceden los 3 MB.

Para poder mantener la funcionalidad de réplica de ficheros debe existir un servicio FTP centralizado o en topología de estrella. Dicho servicio, debe mantener conexión directa con todos los nodos donde se encuentra cada instancia de REKO en el escenario de réplica.

Por lo anteriormente expuesto, se pueden caracterizar como situación problemática, en la que se contextualiza esta investigación, las siguientes ideas esenciales:

1. La solución de réplica no posee un mecanismo de respaldo. En caso de fallo del servidor FTP, se anula totalmente el proceso de réplica de ficheros; lo que genera inconsistencias entre los datos de la base de datos y los ficheros indexados a la misma.
2. El módulo para la réplica de ficheros hace uso de un servidor FTP. El mismo responde a una topología de estrella (para los ficheros que exceden los 3 MB de tamaño), por lo que podría entrar en contradicción con las políticas de redes y seguridad informática de los centros, instituciones o gobiernos cuya topología de red sea diferente a la mencionada anteriormente, elemento que propicia el rechazo de contratos con clientes.
3. Para su funcionamiento, el software depende de un servidor FTP, requisito que impacta negativamente en la diversidad de escenarios en los cuales se pueda instaurar REKO como solución para la réplica de datos.

Por lo anteriormente expuesto se define como **problema científico de la investigación:** ¿Cómo garantizar la fiabilidad, calidad y consistencia de los datos para el software REKO, a la hora de replicar fichero en un escenario de réplica?

Se define como **objeto de estudio** el proceso de réplica de datos en los replicadores de datos, y el **campo de acción** definido en la investigación, se enmarca en el proceso de réplica de ficheros en el Replicador de datos REKO.

Para darle solución a la problemática planteada se define como **objetivo general de la investigación:** desarrollar un componente que permita la replicación de ficheros, el cual garantice la fiabilidad, calidad y consistencia de los datos en un escenario de réplica para el software REKO.

Para dar cumplimiento al objetivo de la investigación se proponen las siguientes **tareas de la investigación:**

- Elaboración del marco teórico de la investigación y análisis del estado del arte sobre las tendencias actuales en la utilización de sistemas para la réplica de ficheros entre bases de datos.
- Selección de la metodología que guiará el proceso de desarrollo de software, descripción de las herramientas, tecnologías, así como el lenguaje que se empleará para el desarrollo del componente de réplica de ficheros en el Replicador de datos REKO.
- Definición de los requisitos funcionales y no funcionales para identificar las capacidades que deben ser alcanzadas por el sistema para cumplir los objetivos trazados.
- Descripción de la arquitectura que soporta la implementación de las funcionalidades.
- Realización del análisis y diseño del componente para describir los artefactos.
- Implementación del componente que llevará a cabo el proceso de réplica de ficheros. Para el desarrollo del mismo, se empleará la metodología y herramientas seleccionadas para la obtención de un producto acorde a las necesidades planteadas.
- Validar la solución propuesta mediante la aplicación de pruebas de software para el aseguramiento de la calidad del producto obtenido.

Con el correcto desempeño de la investigación, se logrará desarrollar un componente para la réplica de fichero en el Replicador de datos REKO. Se eliminará la dependencia de REKO, del servicio FTP para la réplica de ficheros. Se podrá desplegar el software como solución de réplica, sin importar las políticas de redes y seguridad informática implantadas en la institución.

Con el desarrollo del componente se abrirán las puertas a nuevos clientes para la universidad, al poder darle solución a dos de los inconvenientes con los que contaba el replicador anteriormente, los cuales son: la dependencia del replicador del servicio FTP para la réplica de ficheros; así como los conflictos con las políticas de redes y seguridad informática donde se tenía previsto desplegar REKO como solución de réplica, aportando al desarrollo económico del país. La universidad contará con una herramienta mejorada

para la réplica de datos, que puede ser empleada en los proyectos que se desarrollen en la misma.

Para el desarrollo de la investigación se emplea como tipo de estudio el Exploratorio: se utiliza para familiarizarse con el proceso de réplica de ficheros que se emplea actualmente en el Replicador de datos REKO, con el objetivo de poder entender su funcionamiento.

La **idea a defender** planteada en la investigación es**:** el desarrollo de las nuevas funcionalidades a incorporar en el Replicador de datos REKO, permitirá mejorar el proceso de réplica de ficheros, con el que se garantizará la fiabilidad, calidad y consistencia de los datos en un escenario de réplica.

Los métodos de investigación empleados en la presente investigación se listan y describen a continuación:

Métodos teóricos:

- **Histórico-lógico:** su uso permitirá mediante un minucioso estudio, conocer el funcionamiento de la réplica fichero en el Replicador de datos REKO.

- **Analítico-Sintético**: se empleará profusamente en el proceso de análisis documental y revisión bibliográfica, identificándose en virtud de ello, las ideas esenciales que permitirán fundamentar desde el punto de vista teórico la investigación, así como la propuesta que se realiza.

- **Modelación**: se empleará para realizar un diseño simplificado de la realidad a través de los diagramas de clases y de componentes.

Métodos empíricos:

- **Observación científica:** se utilizará con el objetivo de conocer la realidad del impacto de la investigación en el proceso de réplicas de ficheros.

El presente informe de investigación está estructurado de la siguiente manera:

En el **CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO** se realiza un estudio relacionado con los principales conceptos abordados durante la investigación. Se efectúa un análisis de las principales herramientas utilizadas con dicho fin, y se explican las

herramientas así como la metodología seleccionada para efectuar la implementación del sistema.

En el **CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO** se describen las características de la solución propuesta, la especificación y descripción de los requisitos de software. Se describe la propuesta de solución para el problema definido en el diseño de la investigación; y especifican los elementos del diseño como la arquitectura del sistema y los patrones de diseño utilizados.

En el **CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO** se describen los diagramas de componentes y las tareas del ingeniero propuesta, para darle solución a las historias de usuarios definidas en la investigación. Se describen las pruebas realizadas al código, para garantizar la robustez de la solución propuesta, así como los elementos que se tuvieron en cuenta en la implementación del componente.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO

Una vez establecido el marco metodológico que sustenta la investigación; analizados los antecedentes, actualidad, importancia y trascendencia del problema planteado; luego de valorar la necesidad social del tema, así como describir la situación problemática, e identificar el problema científico de la investigación, se procede a detallar el marco teórico de la investigación. El mismo, además de orientar conceptualmente la investigación, permite a partir de la consulta de diferentes fuentes bibliográficas, analizar y exponer aquellas teorías, enfoques teóricos, e investigaciones que se consideren válidos en el contexto del problema planteado.

1.1 Sistema categorial empleado en la investigación

En esta sección se ofrecen las definiciones, así como acercamientos teóricos utilizados en el decurso investigativo para el mejor entendimiento y comprensión de la investigación.

Base de datos

Las bases de datos (BD), son un conjunto de datos organizados, que tienen una estructura lógica y están relacionados entre sí, a los cuales se puede acceder con facilidad con diferentes propósitos. (1)

Una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada. (2)

Una base de datos es un almacén o sistema formado por un conjunto de datos pertenecientes a un mismo contexto, almacenados de forma organizada y relacionada entre sí, permite almacenar información para un posterior uso. (3)

Una base de datos es un sistema informático a modo de almacén. En este almacén se guardan grandes volúmenes de información. Permite automatizar el acceso a la información así como poder acceder a esta de forma rápida y fácil, además de poder efectuar cambios de manera más eficiente. (4)

El autor de la presente investigación, en correspondencia con lo planteado por los líderes de opinión citados anteriormente, se afilia al criterio de que una BD es un conjunto de

datos organizados que tienen una estructura lógica y mantienen una relación entre sí. Una BD está constituida por tablas, relaciones, además de funciones o procedimientos almacenados, donde los datos se almacenan en filas o tuplas, mientras que sus columnas, representan los atributos. Cada atributo se define con un tipo de dato determinado.

Bases de datos distribuidas

Colección de múltiples bases de datos, lógicamente interrelacionadas, distribuidas en la red de ordenadores. (5)

Carrazana Atho define por BD distribuida, un conjunto de múltiples bases de datos lógicamente relacionadas, las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, los cuales tienen la capacidad de procesamiento autónomo, lo cual indica que puede realizar operaciones locales o distribuidas. (6)

En este informe se ha asumido el criterio de Carrazana Atho, que coincide con el acercamiento realizado por K.S. Kalon, atendiendo a que las propuestas de ambos, entroncan con el objetivo general de esta investigación. En la figura 1 se puede observar un ejemplo de base de datos distribuidas.

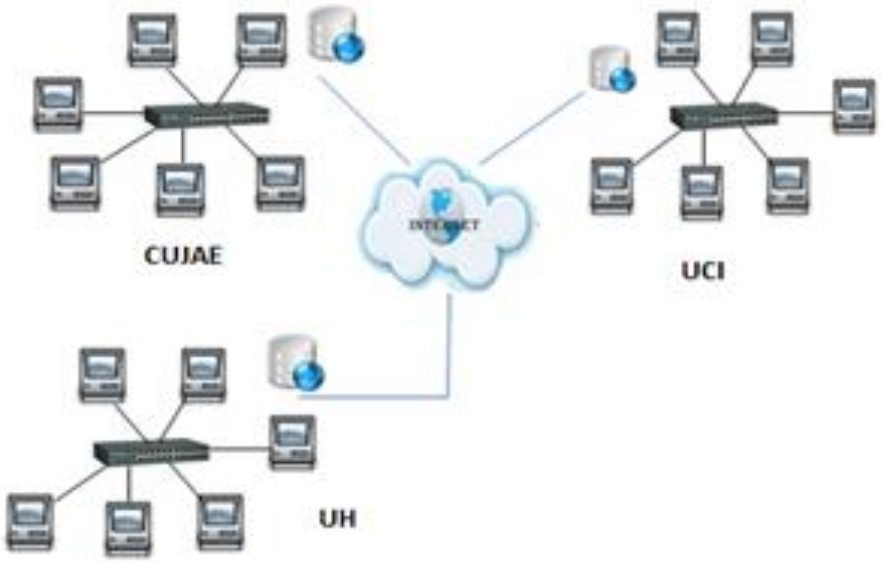


Figura 1 Base de datos distribuida

Fuente: elaboración propia

Replicación

Urbano define por la replicación, en el contexto de los Sistemas de Gestión de Bases de Datos, el proceso de copiar y mantener objetos de una base de datos, entiéndase tablas, secuencias y/o *triggers*, etcétera, en múltiples bases de datos, de tal forma que los cambios realizados localmente, sean enviados a las bases de datos remotas y luego sean aplicados. (7)

Mazilu define por replicación, el proceso de intercambio de información para garantizar la coherencia entre los recursos redundantes, tales como componentes de software o hardware. Permite mejorar la confiabilidad, tolerancia a fallos y accesibilidad. (8)

La replicación es un conjunto de tecnologías destinadas a la copia así como la distribución de datos y objetos de base de datos desde una base de datos a otra, para luego sincronizar ambas bases de datos y mantener su coherencia. (9)

En este trabajo se empleará la definición dada por Randy Urbano de replicación, debido que considera que está actualizado, formalmente fundamentado y acorde a la temática abordada.

Lenguaje de Manipulación de Datos

El Lenguaje de Manipulación de Datos (DML, por sus siglas en inglés), es el lenguaje especializado en la utilización de la base de datos, es decir, permite la consulta y mantenimiento de la base de datos. (2)

Para realizar las consultas a las bases de datos se emplean las siguientes sentencias:

- *Select*: utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
- *Insert*: utilizado para cargar lotes de datos en la base de datos en una única operación.
- *Delete*: utilizado para eliminar registros de una tabla de una base de datos.
- *Update*: utilizado para modificar los valores de los campos y registros especificados. (10)

Componente de software

Aoyama define que un componente separa la interfaz de su aplicación y oculta los detalles de implementación, por lo tanto permite su composición sin la necesidad de conocer los detalles de implementación de componentes. (11)

Szyperski define que un componente es una unidad de composición que puede ser desplegado de forma independiente y está sujeto a la composición de terceras partes. (12)

Crnkovic define que un componente es una unidad reutilizable de despliegue y composición al que se accede a través de una interfaz. Una interfaz especifica los puntos de acceso a un componente. (13)

Sommerville define que un componente es una entidad ejecutable independiente. Los servicios ofertados por un componente están disponibles a través de una interfaz. La interfaz del componente se expresa en términos de operaciones parametrizadas y su estado interno nunca se muestra. (14)

En el decurso investigativo de este trabajo se ha aceptado como válida la definición brindada por Sommerville, debido a que se ha considerado que está formalmente fundamentada y acorde a la temática abordada.

Conceptos clave de las variables definidas en la investigación

Las definiciones de las palabras que se listan a continuación, son las empleadas por la Real Academia de la Lengua Española.

- Fiabilidad.
- Calidad.
- Consistencia.

Se define por fiabilidad:

- Probabilidad de buen funcionamiento de algo.
- Cualidad de fiable.

Se define por fiable:

- adj. Dicho de una persona: Que es digna de confianza.
- adj. Que ofrece seguridad o buenos resultados. Mecanismo fiable. Método fiable.
- adj. Creíble, fidedigno, sin error. Datos fiables.

Se define por calidad:
Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.

Se define por consistencia:

- Duración, estabilidad, solidez.
- Trabazón, coherencia entre las partículas de una masa o los elementos de un conjunto. (15)

El autor de la presente investigación, se afilia al criterio de que la fiabilidad es el buen funcionamiento de algo, ofrece seguridad, o buenos resultados. El principal criterio para validar la fiabilidad, es que el componente a desarrollar deberá garantizar, en todas las instancias de REKO, presentes en el escenario de réplica, el fichero ha sido reconstruido satisfactoriamente.

En este informe, se define por calidad la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. El componente para la réplica de ficheros, deberá llevar a cabo el proceso de replicación de fichero satisfactoriamente, sin necesidad de hacer uso del servicio FTP.

En correspondencia con la Real Academia de la Lengua Española se define por consistencia, la coherencia de los elementos de un conjunto. El componente a desarrollar, deberá afirmar la coherencia entre los paquetes enviados, donde se garantice que ninguno de ellos cuente con información corrupta. El hecho de no contar en ningún paquete con información corrupta, asegurará la reconstrucción satisfactoria del fichero en el nodo destino. El fichero reconstruido en el nodo receptor, coincidirá plenamente con el fichero enviado desde el nodo emisor.

1.2 Análisis de las soluciones existentes

Un número de empresas comercializan soluciones de réplica en el mercado del software; algunas incluidas dentro del software y otras, como opciones que se pueden licenciar de forma independiente. En el proceso laborioso de búsqueda de una solución para la réplica de ficheros en el replicador REKO, se realizó un estudio de replicadores con características similares y que a continuación se exponen:

1.2.1 Daffodil Replicator v2.1

Es una herramienta implementada sobre Java, de código abierto para la integración, migración y protección de datos en tiempo real. Permite bi-direccionar datos de replicación así como sincronización entre bases de datos homogéneas y heterogéneas; se incluye Oracle, MySQL, Daffodil DB, PostgreSQL, entre otras. Incluye además:

- La capacidad de detectar y permitir resolución de conflictos.
- Independencia de la plataforma.
- Soporte para la replicación de datos de gran tamaño (Clob, Blob).

La principal limitante detectada en el estudio de este replicador es que no es capaz de replicar en un escenario multi-maestro. (16)

1.2.2 Tungsten Replicator v2.0

Es un sistema *Open Source* de replicación maestro-esclavo. Su modo de replicación es basado en *logs*. A continuación se listan y describen sus principales características:

- Soporte para replicación entre bases de datos heterogéneas.
- Bases de datos soportadas son MySQL, Oracle y PostgreSQL.
- Es independiente de la plataforma.
- Replica entre bases de datos con estructuras iguales.
- Replica archivos de gran tamaño.

Pese a sus características, este replicador no es capaz de replicar en un escenario multi-maestro y la forma de captura y almacenamiento de los cambios es mediante *logs*, lo que constituye otra limitante. (17)

1.2.3 DBReplicator

Es un software de replicación multi-maestro, abierto para la red de aplicaciones Java. Las principales características de DBReplicator son las siguientes:

- Principales servidores de BD soportados: MySQL, Oracle, PostgreSQL, Microsoft SQL Server.
- Sincronización bi-direccional.
- Independiente de la plataforma.
- Detección y resolución de conflictos.
- Tareas programadas.
- Codificación de caracteres que no estén en formato ASCII en el momento que se crean archivos de sincronización en XML.
- Creación de tablas en el subscriptor, si la tabla que replica desde el publicador no está presente en la base de datos del subscriptor.

A pesar de las características mencionadas, tampoco responde a las necesidades planteadas, pues no es capaz de replicar datos de gran tamaño (Clob, Blob, Bytea, etc.). (18)

1.2.3 SymmetricDS

Es un sistema de software libre, escrito en Java, liberado bajo los términos LGPLi. Es una solución de sincronización que muestra las siguientes características:

- Brinda una interfaz web para la configuración de la réplica.
- Soporta los gestores MySQL, Oracle, SQL Server, PostgreSQL, DB2, Informix, Interbase, Firebird, HSQLDB, H2, Apache Derby y Greenplum.
- Basa su replicación en disparadores que capturan los cambios en la base de datos.
- Puede instalarse como una aplicación web independiente o ser embebida dentro de una aplicación desarrollada en Java.
- Fue diseñado a escala para un gran número de bases de datos, para trabajar a diez a través de conexiones de baja velocidad, y soportar los períodos de interrupción de la red.

- En la versión 3.1.9 incluyen la agrupación del trabajo de depuración y mejoras en el rendimiento de los datos de encaminamiento para un gran número de nodos.

Aunque hace uso del modo asíncrono como forma de transmisión de los cambios, y el uso de *triggers* para la captura y almacenamiento de los mismos, no cumple con los requerimientos planteados, pues no es capaz de replicar datos de gran tamaño (Clob, Blob, Bytea, etc.). (19)

1.2.4 Replicador de datos REKO

El Replicador de datos REKO constituye una solución práctica y de bajos esfuerzos de configuración a las principales necesidades de replicación en la mayoría de los sistemas actuales. Como se observa en la figura 2, REKO permite que dos BD puedan replicar entre sí, aunque se encuentren en subredes distintas y utilicen protocolos de transmisión diferentes, la única restricción es que todos deben estar conectados a un servidor JMS central, o en caso de un clúster de servidores JMS, estar conectados a uno de ellos.

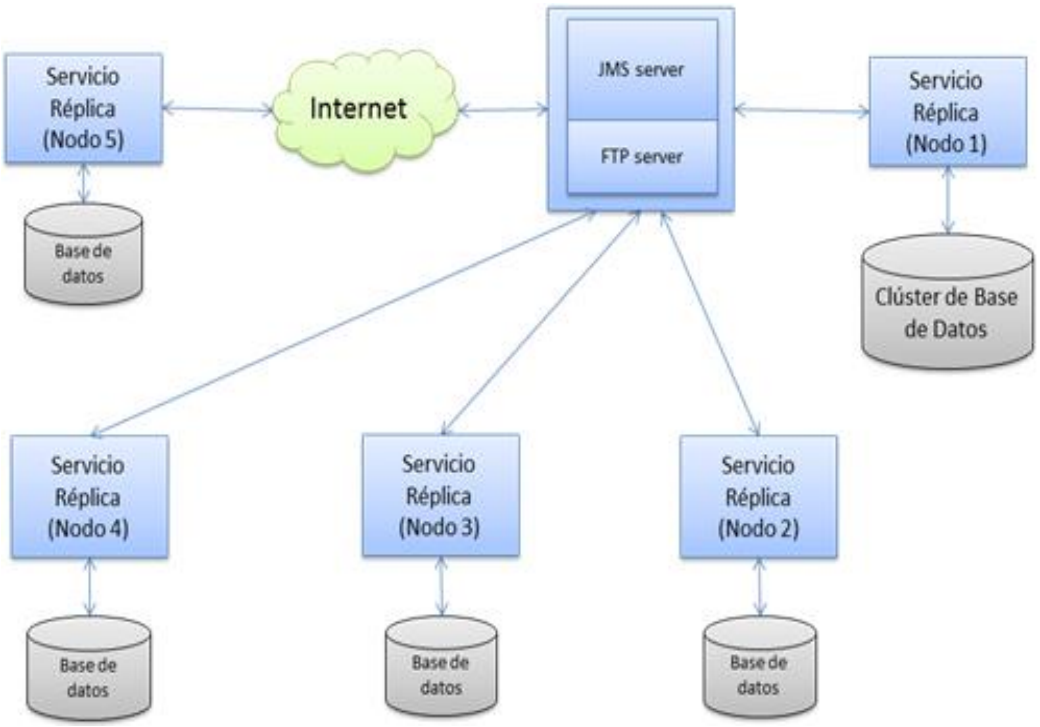


Figura 2 Ejemplo de escenario de replicación en REKO

Haciendo uso de la metodología de desarrollo SXP, el lenguaje unificado de modelado UML, la herramienta CASE Visual Paradigm v6.4, Java como lenguaje de programación y

Eclipse v3.2.0, como entorno integrado de desarrollo, fue estudiado con profundidad su funcionamiento así como valoradas y ejecutadas las modificaciones pertinentes en componentes específicos; así como, la adición de otros; lo que permitió evaluar su funcionamiento con el servidor de mensajería ActiveMQ y el servidor FTP ProFTPd.

Su desarrollo sobre la plataforma de Edición Empresarial de Java (JEE, por sus siglas en inglés), permite que pueda ser ejecutado en cualquier sistema operativo. Se utilizó un diseño desacoplado que ha permitido extender con poco esfuerzo sus funcionalidades. Se empleó, además, el *framework* Spring (19, 20) .

La transferencia de datos de replicación puede realizarse a soporte para replicación sobre el Protocolo de Control de Transmisión (TCP/IP, por sus siglas en inglés), FTP y el Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés). Los archivos de gran tamaño son enviados por FTP, lo que permite resumir la trasmisión en caso de interrupciones en la red. La administración y configuración de la réplica se realiza a través de una interfaz Web, por lo que es administrable vía remota exclusivamente a través de un navegador.

Principales funcionalidades que brinda el software REKO en su versión 3.0:

- Soporta la replicación de transacciones a través de Hibernate, JDBC y la replicación de acciones a través de *triggers*.
- Soporte para réplica de datos en ambientes conexión y sin conexión.
- Soporte para réplica entre bases de datos con diferentes estructuras.
- Soporte para réplica de ficheros externos la base de datos.
- Soporte para la Sincronización de datos en ambientes con conexión y sin conexión.
- Selección de los datos de réplica ajustada por filtros.
- Soporte para réplica de datos entre los gestores PostgreSQL, Oracle, MySQL, y Microsoft SQL Server.
- Soporte para resolución de conflictos automática y manual.
- Monitoreo en tiempo real de los datos de réplica.
- Los archivos de gran tamaño son enviados por FTP, lo que permite resumir el envío y el recibo de los mismos en caso de interrupciones en la red

- Soporte para programación del momento de captura y envío de los Datos.
- Seguridad de los datos que se envían con encriptación SSL. (20)

1.3 Resultados

Al concluir el estudio se pudo constatar que ninguno de estos replicadores respondía a las necesidades actuales de REKO, sin embargo respondían algunas de ellas por separado.

DaffodilReplicator y TungstenReplicator no son capaces de replicar en un escenario multi-maestro, DbReplicator y SymmetricDS, no son capaces de replicar ficheros.

Tabla 1 Resultados del análisis de los replicadores de datos

Especificaciones/ Herramientas	Daffodil Replicator	SymmetricDS	Tungsten Replicator	DbReplicator	REKO
Maestro - Esclavo	x		x		
Multi - Maestro		x		x	x
PostgresSQL	x	x	x	x	x
MySQL	x	x	x	x	x
Oracle	x	x	x	x	x
Microsoft SQLServer		x		x	x
Otros gestores		x			

Fuente: elaboración propia a partir de (16-21).

Por lo anteriormente planteado, se decide por parte del autor de la presente investigación, dedicar tiempo y esfuerzo, al desarrollo de un componente para la réplica de ficheros, capaz de responder a las necesidades actuales de REKO.

1.4 Metodología de desarrollo de software

Las metodologías para el desarrollo del software imponen un proceso disciplinado con el fin de hacerlo más predecible y eficiente. Tiene como principal objetivo aumentar la

calidad del software que se produce en todas y cada una de sus fases de desarrollo. Se hace énfasis en la calidad y menor tiempo de construcción del software o lo que es lo mismo “producir lo esperado en el tiempo esperado y con el coste esperado”. (22)

Luego de un estudio realizado por especialistas de la Dirección General de Producción (DGP) en la UCI, se define como política de la universidad, el uso de la metodología Proceso Unificado Ágil, (AUP, por sus siglas en inglés), con modificaciones para el marco de desarrollo de la UCI; para guiar el proceso de desarrollo de software en todos los proyectos productivos que se desarrollan en la universidad. Debido a que la solución propuesta en esta investigación, se encuentra adscrita a uno de los proyectos de desarrollo de software de la UCI, y en aras de cumplir con las políticas definidas por la universidad, se adopta la metodología AUP, para guiar el proceso de desarrollo de software.

Metodología Proceso Unificado Ágil

La metodología AUP, es una versión simplificada del Proceso Unificado de Racional (RUP, por sus siglas en inglés). Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio a través de técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

La metodología AUP aplica técnicas ágiles, donde se incluye:

- Desarrollo Dirigido por Pruebas (TDD, por sus siglas en inglés).
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de BD para mejorar la productividad.

El Proceso Unificado (UP, por sus siglas en inglés), es un marco de desarrollo software iterativo e incremental. A menudo es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software. Dado que es un marco de procesos, puede ser adaptado y la más conocida es RUP de IBM. (23)

La UCI define las fases de Inicio, Ejecución y Cierre para el ciclio de vida de los proyectos. Para una mayor comprensión se muestra la siguiente Tabla 2.

Tabla 2 Fases variación AUP-UCI

Fases	Objetivos de las fases
Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo así como decidir si se ejecuta o no el proyecto.
Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, se incluye el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Fuente: elaboración propia a partir de (24)

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos así como Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas como se muestra en la Tabla 3.

Tabla 3 Disciplinas variación AUP-UCI

Disciplinas	Objetivos de las disciplinas
Modelado de Negocio (opcional)	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
Análisis y Diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, así como una descripción que sea fácil de mantener y ayude a la estructuración del sistema. En esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, e incluya los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Prueba Interna	En esta disciplina se verifica el resultado de la implementación probando cada construcción; se incluye tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.

Prueba de Liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
Prueba de Aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones, así como tareas para las cuales el software fue construido.
Despliegue (opcional)	Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.

Fuente: elaboración propia a partir de : (24)

Descripción de los roles

Para el ciclo de vida de los proyectos de la UCI se definen 11 roles. En la Tabla 4 se describen los roles y artefactos de salida propuestos por la metodología.

Tabla 4 Roles y artefactos de salida AUP-UCI

Roles AUP	Roles Variación AUP-UCI	Artefactos de salida
Administrador de proyecto	Jefe de proyecto Planificador	Registro de proveedores de requisitos. Criterios para validar requisitos del cliente. Criterios para validar requisitos del producto. Estándar de codificación. Plan de desarrollo de software. Método de estimación.
Ingeniero de proceso Modelador ágil	Analista Arquitecto de información (Opcional)	Registro de proveedores de requisitos. Evaluación de requisitos. Requisitos rechazados. Criterios para validar requisitos del cliente. Criterios para validar requisitos del producto. Historias de usuario. Modelo conceptual. Especificación de

		requisitos de software. Manual de usuario. Plan de desarrollo de software.
Desarrollador	Desarrollador	Estándar de codificación.
Administrador de la configuración	Administrador de la configuración	Plan de desarrollo de software.
Stakeholder	Stakeholder (Cliente/Proveedor de requisitos)	Método de estimación. Solicitud de oferta de productos y de servicios.
Administrador de pruebas	Administrador de calidad	Diseño de casos de prueba. Manual de usuario. Plan de pruebas. Plan de desarrollo de software.
Probador	Probador	Diseño de casos de prueba.
Administrador de BD	Arquitecto de software (Sistema) Administrador de BD	Modelo de diseño. Arquitectura de software.

Fuente: de elaboración propia a partir de (24)

Principios de la AUP

- El personal tiene conocimiento de la actividad que realiza. La gente no va a leer con lujo de detalles el proceso de documentación, pero algunos quieren una orientación de alto nivel y/o formación cada cierto intervalo de tiempo. La AUP producto proporciona enlaces a muchos de los detalles, si usted está interesado, pero no obliga a aquellos que no lo deseen.

- Simplicidad: todo se describe concisamente a través de un puñado de páginas, no miles de ellas.
- Agilidad.
- Centrarse en actividades de alto valor. La atención se centra en las actividades que son esenciales para el desarrollo, no todas las actividades que suceden forman parte del proyecto.
- Herramienta de independencia. Usted puede usar cualquier conjunto de herramientas que desee con el ágil UP. Lo aconsejable es, utilizar las herramientas que son las más adecuadas para el trabajo, que a menudo son las herramientas simples, o incluso herramientas de código abierto.
- Adaptación de este producto para satisfacer sus propias necesidades. La AUP producto es de fácil acomodo común a través de cualquier herramienta de edición de HTML. No se necesita comprar una herramienta especial, o tomar un curso, para adaptar la AUP.

Ventajas de AUP

- El personal sabe lo que esta hace: no obliga a conocer detalles.
- Simplicidad: apuntes concisos.
- Agilidad: procesos simplificados del RUP
- Centrarse en actividades de alto valor: esenciales para el desarrollo.
- Herramientas independientes: a disposición del usuario.
- Fácil adaptación de este producto: de fácil acomodo (HTML).

Desventajas de AUP

- El AUP es un producto muy pesado en relación al RUP.
- Como es un proceso simplificado, muchos desarrolladores eligen trabajar con el RUP, por tener a disposición más detalles en el proceso.(23)

1.5 Herramientas y lenguajes para el desarrollo

El estudio del entorno de réplica de datos del software REKO, permitió realizar el análisis de los lenguajes y herramientas que han sido empleadas desde su versión 1.0. Las mismas continuarán siendo utilizadas puesto que se disminuye la curva de aprendizaje y el tiempo de desarrollo.

1.5.1 Herramienta para la Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés)

Visual Paradigm para UML 8.0: herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. Cuenta con la ventaja de ser una herramienta multiplataforma y permite la integración con Eclipse IDE. (25)

Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar, es compatible entre ediciones. Genera código para Java y exportación como HTML. (26)

1.5.2 Plataforma de desarrollo

Edición Empresarial de Java (JEE, por sus siglas en inglés): es un entorno independiente de la plataforma centrado en Java para desarrollar, crear e implementar en línea aplicaciones empresariales basadas en web. Consta de un conjunto de servicios, Interfaces de Programación de Aplicaciones (APIs, por sus siglas en inglés) y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones basadas en web de varios niveles. (27)

1.5.3 Lenguaje de programación

Java 7.0: lenguaje de programación orientado a objeto, de código abierto e independiente de la plataforma. Se distingue por ser capaz de gestionar la memoria automáticamente, posee mecanismos de seguridad incorporados, los cuales limitan el acceso a recursos de las máquinas donde se ejecuta e incorpora herramientas de documentación. Es multihilos. Todas estas características lo califican como un lenguaje de programación robusto.(28)

1.5.4 Lenguaje de modelado

UML 2.0: es un lenguaje estándar en el análisis y diseño de sistemas de cómputo que permite especificar, visualizar, construir así como documentar todos los elementos que forman un sistema de software, desde una perspectiva orientada a objetos. Este modelado visual es independiente del lenguaje de implementación, los diseños realizados a través de UML, se pueden implementar en cualquier lenguaje que soporte las posibilidades de UML, principalmente lenguajes orientados a objetos. (29)

1.5.5 Tecnologías empleadas

Spring 2.5: contenedor y *framework* de peso ligero, basado en inyección de dependencias y orientación a aspecto, lo que promueve un bajo acoplamiento pues los objetos reciben pasivamente sus dependencias en lugar de crearlas o buscar los objetos dependientes por sí mismos. Brinda facilidades para desarrollar una aplicación empresarial en JEE. (30)

Servicio de Mensajería Java (JMS, por sus siglas en inglés): estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma JEE crear, enviar, recibir y leer mensajes. Hace posible la comunicación confiable de manera síncrona y asíncrona. (31)

Dojo 1.4: framework que posee Interfaz de Programación de Aplicaciones (APIs) y Widgets (pequeña aplicación o programa) para facilitar el desarrollo de aplicaciones Web modernas. Tiene como características principales que es proyecto Open Source, presenta una multitud de componentes visuales que permiten el desarrollo rápido de interfaces de usuario complejas, tiene múltiples implementaciones para acceso a datos, incluido JSON-Rest y permite importar estilos Cascading Style Sheets (CSS). (43)

Herramientas utilizadas

Apache ActiveMQ 5.9.0: es el más popular y poderoso cliente de mensajería de código abierto escrito en Java. Fue diseñado para comunicarse a través de una serie de protocolos con el apoyo de diferentes lenguajes de programación tales como: Java, C++, C# entre otros. (32)

Apache Tomcat 7.0: contenedor de *servlet* usado en la implementación de referencia oficial para las tecnologías Java Servlet y Java Server Pages. Es desarrollado en un ambiente participativo y abierto. (33)

JUnit 4.4: framework de código abierto para diseñar, construir y ejecutar pruebas unitarias en Java. Se integra con muchos IDEs como Eclipse. Con JUnit se logra ejecutar pruebas automatizadas fácilmente. Contiene mecanismos para recolectar resultados de manera estructurada, produce varios tipos de reportes a partir de los resultados obtenidos en la ejecución de las pruebas, permite que existan relaciones entre las pruebas y que

unas reutilicen código de otras. Además, soporta la jerarquía entre las pruebas, lo que permite establecer la prioridad y el orden de unas con otras. (34)

1.5.6 Entorno de desarrollo integrado

Eclipse STS v3.2: es una herramienta libre desarrollada por Spring Source, basada actualmente en Eclipse 3.x, para construir aplicaciones empresariales enriquecidas por los proyectos de Spring Source. Una de las principales razones para emplear STS es que incluye herramientas para el desarrollo en lenguaje Java, para Spring. (35)

1.5.7 Sistema de gestión de base de datos

PostgreSQL: gestor de bases de datos orientadas a objetos, muy conocido y usado en entornos de software libre. Puede funcionar en múltiples plataformas. Se empleará una versión de PostrgreSQL8.4 o superior (36)

SQLServer: amplia plataforma de base datos de Microsoft, ofrece un rendimiento fiable gracias a la integración de tecnologías en memoria, una rápida obtención de información útil a partir de cualquier tipo de datos, con herramientas que conocemos, como Excel, y una plataforma para compilar, implementar así como administrar soluciones tanto locales como en nube. (37)

Oracle Database 12c: presenta una nueva arquitectura, facilita la rápida consolidación de muchas bases de datos y su gestión como un servicio de nube. Oracle Database 12*c* incluye también capacidades de procesado de datos en memoria que ofrecen un rendimiento analítico revolucionario. Otras innovaciones de la base de datos ofrecen nuevos niveles de eficacia, rendimiento, seguridad y disponibilidad. Oracle Database 12*c* se presenta en tres ediciones adaptadas a sus necesidades de negocio y su presupuesto: Enterprise Edition, Standard Edition y Standard Edition One. (38)

MySQL: representa el SGBD de código abierto más popular del mundo debido a que constituye una completa herramienta y es capaz de acumular una enorme cuantía de datos de gran diversidad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización. Es completamente gratuito para gran parte de sus servicios, dispone de muchas de las funciones que exigen los desarrolladores profesionales, es multiplataforma ya que se ejecuta en la mayoría de los sistemas operativos y resulta fácil de utilizar así

como de administrar. Actualmente MySQL compite con SGBD propietarios como Oracle, SQL Server y Db28. (39)

1.5.8 Librería de programación de aplicaciones

Berkeley DB 12c: familia de bibliotecas para bases de datos embebidas clave-valor que prestan servicios de gestión de datos de alto rendimiento escalables a las aplicaciones. Utilizan APIs para las funciones de llamada en el acceso y la gestión de datos.(40)

1.6 Consideraciones parciales del capítulo

Teniendo en cuenta lo anteriormente planteado se han arribado a las siguientes consideraciones parciales:

- No es viable la utilización total o parcial de herramientas de replicación ya desarrolladas para la solución del problema, debido a que ninguno de estos replicadores responden a las necesidades actuales de REKO, sin embargo responden algunas de ellas por separado, Daffodil Replicator y TungstenReplicator no son capaces de replicar en un escenario multi-maestro, DbReplicator y SymmetricDS no son capaces de replicar ficheros. Por lo que se decide, por parte del equipo de desarrollo de REKO, dedicar tiempo y esfuerzo, al desarrollo de un componente para la réplica de ficheros, capaz de responder a las necesidades actuales de REKO.
- El proceso desarrollo de software, será guiado por la metodología AUP, por política de la universidad.
- Para el desarrollo del componte, se harán uso de las tecnologías y herramientas empleadas para el desarrollo REKO, por política de proyecto.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO

Finalizada la fundamentación teórica de la investigación se procede a caracterizar el sistema para planificar, organizar y ejecutar el desarrollo de la solución. Para guiar el proceso de desarrollo de software, se hace uso de la metodología AUP.

2.1 Funcionamiento de REKO

Cada cierto intervalo de tiempo, el Capturador de Cambios de REKO registra los cambios realizados sobre la base de datos y crea un objeto de tipo ReplicableGroup (grupo replicable). Cada objeto creado contiene un grupo de objetos de la clase ReplicableAction (acción replicable), donde cada uno almacena toda la información necesaria para replicar los cambios. A partir de estos objetos, las configuraciones registradas y los filtros asociados a estas, se determinan los destinos hacia donde se enviará el grupo replicable. De la propagación de los grupos replicables se encarga el Distribuidor de Datos a través del servidor JMS.
En cada nodo destino, una vez recibido un grupo replicable, pasa al Aplicador de Cambios para ejecutar en la base de datos del destino las acciones que contiene.
Existen mecanismos de chequeo de envío, confirmación de recepción y registro en una base de datos local, de los grupos replicables, que garantizan la integridad, así como la persistencia de los datos que se replican, ante posibles eventualidades. (41)

El software REKO, se ejecuta en un entorno de replicación multi-maestro teniendo una instancia en cada nodo, los mismos pueden estar agrupados por etiquetas. Desde cada nodo se permite configurar la réplica hacia otro nodo o hacia una etiqueta; además de poder realizarse distintas configuraciones según los nodos y etiquetas que se definan.
Las configuraciones independientes de las tablas que se desean replicar son registradas por configuración. Cada una de estas tablas puede ser configurada para replicar en dependencia de la acción que se efectúe sobre ella (inserción, actualización, eliminación), además pueden definirse filtros SQL para determinar por cada acción si se replicará o no el cambio. Para el manejo de fichero, se define en la configuración de réplica, la columna donde se almacenará la referencia del fichero externo. Se especifica el nombre del fichero que se referenciará.

El módulo para la réplica de ficheros, cuyo funcionamiento se basa en el uso de un servidor FTP para los archivos que su peso supere los 3MB. Los ficheros que no exceden los 3 MB se

envían a través del servidor ActiveMQ. Para poder mantener la funcionalidad de réplica de ficheros, debe existir un servicio de FTP centralizado, o en topología de estrella. Dicho servicio debe mantener conexión directa con todos los nodos donde se encuentra cada instancia de REKO en el escenario de réplica. El módulo para la réplica de ficheros cuenta con los siguientes inconvenientes:

- Mecanismo de respaldo: el módulo para la réplica de ficheros, no cuenta con un mecanismo de respaldo en caso de fallo del servicio FTP, deja de funcionar totalmente el proceso de réplica de ficheros.
- Dependencia de servicios terceros: dependencia de REKO para su funcionamiento del servicio tercero FTP.
- Topología de red: el módulo para la réplica de ficheros está implementado en topología estrella. El mismo puede entrar en contradicción con las políticas de redes y seguridad informática de la institución donde se desplegará el sistema con topología diferente a la implementada en el módulo.

Ante estas limitantes, se requiere modificar en REKO el módulo de réplicas de ficheros, añadiéndole un componente capaz de garantizar la fiabilidad, integridad y consistencia de los datos en un escenario de réplicas para el software REKO.

2.2 Modelo de dominio

Un modelo de dominio captura los tipos de objetos más importantes en el entorno del sistema. Los objetos o clases del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que el sistema trabaja. Se describe mediante diagramas de UML que muestran las clases del dominio y como se relacionan unas con otras mediante asociaciones. (42)

2.2.1 Diagrama de clases del modelo de dominio

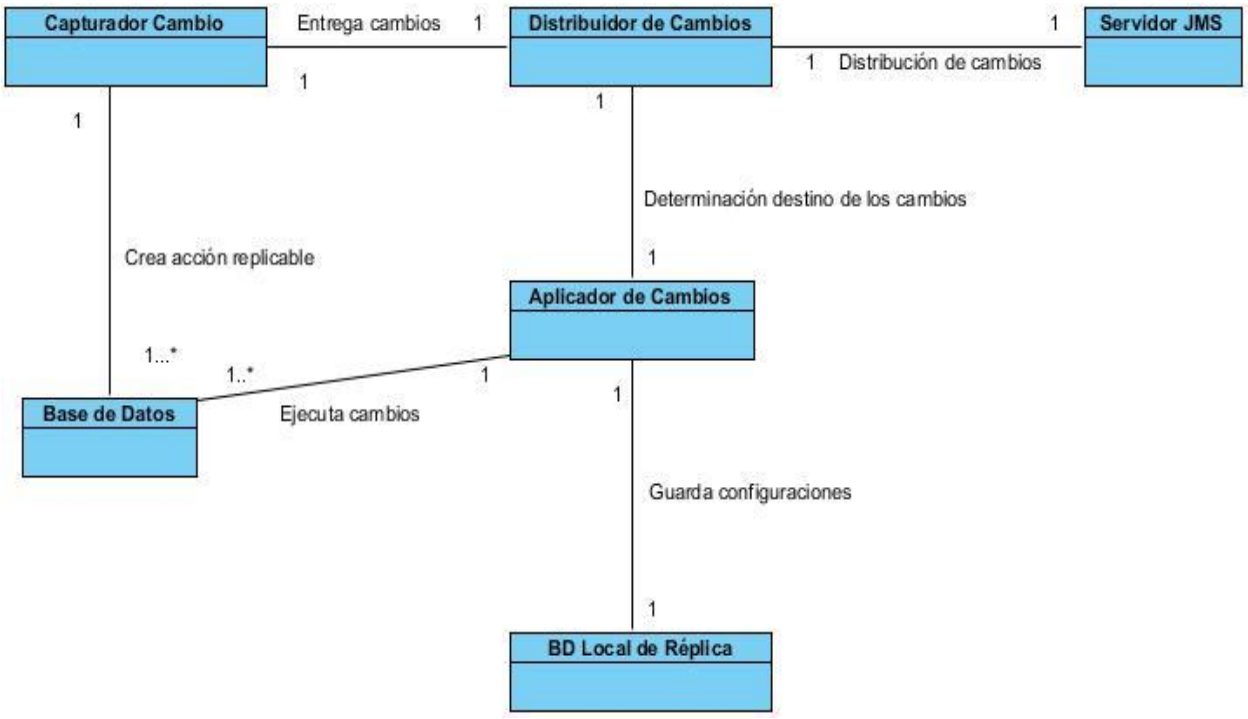


Figura 3 Modelo de dominio

2.2.2 Descripción de las clases del modelo de dominio

A continuación se listan y describen a grosso modo las clases presentes en la figura 4 del modelo de dominio propuesto:

Capturador de Cambios: Captura los cambios que se realizan sobre la BD y se los entrega al Distribuidor de Cambios.

Aplicador de Cambios: Ejecuta sobre la BD los cambios que sean replicados hacia la BD.

Distribuidor de Cambios: Determina para dónde debe ser enviado cada cambio realizado en la BD, los envía y se responsabiliza de que cada cambio llegue a su destino.

BD Local de Réplica: Es utilizada para guardar las configuraciones propias de la réplica, las acciones sobre la BD que han dado conflicto al aplicarse y las acciones o transacciones que no han podido llegar a su destino.

Servidor JMS: Servidor de Mensajería Java (JMS, por sus siglas en inglés), bajo la especificación Java Message Service, es utilizado como punto intermedio en la distribución de la información enviada bajo JMS. (21)

2.3 Requisitos funcionales

Los requisitos funcionales de un sistema describen las capacidades o funciones que el sistema debe cumplir, los servicios que de él se esperan, o los que proveerá. (51) .

Los requisitos funcionales (RF), identificados se listan y describen a continuación:

Tabla 5 Descripción de los RF.

No	Nombre	Descripción	Prioridad para el cliente	Complejidad
RF1	Enviar fichero	Permitirá enviar un fichero hacia todos los nodos destinos registrados.	Alta	Alta
RF1.1	Obtener fichero de ReplicableAction.	Permitirá obtener el objeto de tipo File, indexado dentro de la acción replicable (ReplicableAction).	Alta	Alta
RF1.2	Particionar fichero obtenido.	Permitirá fragmentar un fichero en pequeños segmentos en dependencia del tamaño del buffer definido en la configuración general de la aplicación. Crea objetos de tipo PackageSend para cada fragmento.	Alta	Alta
RF1.3	Persistir fichero.	Permitirá luego de ejecutado el RF 1.2, almacenar paquetes creados en una BD embebida local.	Alta	Alta
RF1.4	Enviar paquetes a nodos destinos.	Luego de ejecutado el RF 1.3, el sistema mapeará información almacenada en la BD embebida. Al concluir el mapeo de la información, el sistema dará inicio al proceso de envío de paquete hacia todos los nodos destinos.	Alta	Alta
RF1.5	Recibir notificación de recibo por nodos emisor	El sistema luego de enviar un paquete hacia los nodos receptores, para enviar próximo paquete, espera confirmación de recibo, emitido por el nodo receptor al arribar un paquete al nodo. Al recibir notificación en nodo emisor, da continuidad al proceso de envío de paquete hacia los nodos destinos.	Alta	Media

FR1.6	Enviar notificación de paquete completo al enviar último paquete del fichero particionado	Luego del envío del último segmento del fichero analizado, el sistema procede al envío hacia nodo receptor de notificación de fichero completo.	Alta	Media
RF1.7	Limpiar BD embebida y mapa que contiene información de fichero analizado.	Luego de tener notificación que el fichero enviado fue reconstruido satisfactoriamente en todos los nodos destinos, el sistema procederá a la limpieza de la BD embebida y mapa que contiene información del fichero analizado.	Alta	Alta
RF2	Construir fichero.	El sistema, luego de ejecutado el RF 2.5, procede a la reconstrucción del fichero.	Alta	Alta
RF2.1	Capturar paquete enviado.	El sistema capturará objeto de tipo PackageSend enviado por nodo emisor.	Alta	Media
RF2.2	Verificar si paquete recibido es el esperado.	Luego de ejecutado RF 2.1, el sistema verifica si paquete recibido es el esperado.	Alta	Media
RF2.3	Persistir paquete recibido.	Si el paquete recibido en RF 2.1 es el esperado, el sistema procederá a almacenar el objeto recibido en la BD embebida local.	Alta	Alta
RF2.4	Enviar notificación de paquete recibido a nodo emisor.	Luego de ejecutado los RF 2.1 y RF 2.2, el sistema envía notificación de recibo a nodo emisor, tanto de paquete esperado correcto, como paquete no esperado; se especifica en esta última notificación, el número de paquete esperado en el nodo receptor.	Alta	Media
RF2.5	Recibir notificación de paquete completo.	Luego de ejecutado el RF 1.6, procede a la captura de la notificación de fichero completo.	Alta	Media
RF 2.6	Generar notificaciones de fichero reconstruido.	El sistema, luego de ejecutado el RF 2.0, procede al envío de la notificación de fichero reconstruido en nodo destino.	Alta	Media
RF 2.7	Limpiar la BD embebida	Luego de ejecutado el RF 2.0, se dará inicio al proceso de limpieza	Alta	Alta

	local, que contiene toda la información necesaria del fichero analizado en nodo destino.	de la información almacenada en la BD embebida local.		

2.4 Requisitos no funcionales

Los requisitos no funcionales RNF definen las restricciones del sistema, son propiedades o cualidades que el sistema debe poseer. Representan las características del producto. (51)

Los requisitos no funcionales detectados se listan y describen a grosso modo:

Tabla 6 Descripción de los RNF

Fiabilidad.	Tolerancia a fallos y recuperabilidad: Capacidad del producto para operar según lo previsto en presencia de fallos de hardware o software. (ver Anexo 1) Capacidad del producto para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallos. (ver Anexo 2)
	Madurez: Capacidad del producto para satisfacer las necesidades de fiabilidad en condiciones normales. (ver Anexo 3)
	Disponibilidad: Capacidad del producto de estar operativo y accesible para su uso cuando se requiere. (ver Anexo 4)
Mantenibilidad.	Analizabilidad: Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar. (ver Anexo 5)

	Modificabilidad: Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño. (ver Anexo 6)
	Capacidad para ser probado: Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios. (ver Anexo 7)
Portabilidad.	Adaptabilidad: Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales, o de uso. (ver Anexo 8)
	Reemplazabilidad: Grado en que un producto puede sustituir a otro producto de software especificado para el mismo propósito en el mismo entorno. (ver Anexo 9)
Adecuación funcional.	Integridad funcional: Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificados. (ver Anexo 10)
Seguridad.	No repudio: Grado en que las acciones o eventos pueden ser probados a haber tenido lugar, por lo que los eventos o acciones no pueden ser repudiados más tarde. (ver Anexo 11)
Eficiencia en el rendimiento.	Comportamiento en el tiempo: Grado en que los tiempos de respuesta, procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus

	funciones cumplen con los requisitos. Utilización de recursos: Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos. (ver Anexo 12)

Fuente: Elaboración propia (45)

2.5 Propuesta de solución

Al ejecutarse el Capturador de Cambio, se crea un objeto de tipo ReplicableGroup, que es capturado por la clase SenderManager. Esta clase analiza si el objeto contiene un fichero indexado; en caso afirmativo, se obtiene el fichero referenciado, dando inicio al proceso de envío de fichero hacia todos los nodos destinos. En los nodos destinos se capturan los paquetes que se han enviado; y se genera una notificación al nodo emisor del paquete recibido. El paquete obtenido es almacenado en la BD embebida local, de donde se leerán todos los paquetes en el proceso de reconstrucción del fichero, que comienza al capturar la notificación de envío completo en el nodo receptor.

Existen mecanismos de chequeo de envío, confirmación de recepción y registro de los paquetes recibidos en una BD embebida, que garantizan la integridad referencial y persistencia de cada paquete del fichero ante posibles eventualidades. Se garantiza que en todos los nodos registrados, el fichero enviado fue reconstruido satisfactoriamente. Se garantiza la integridad de la información.

2.6 Historias de usuario (HU)

Las HU sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia. (55)

A continuación en las tablas 11 y 12 se describen la principales HU a emplear en el desarrollo del componente.

Tabla 7 HU Enviar fichero

Número: HU1	**Nombre del requisito:** Enviar Fichero
Programador: Walter Jacas Jardines	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3 semanas
Riesgo en desarrollo: • Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas eléctricos.	**Tiempo real:** 3 semanas
Descripción: Al ejecutarse alguna operación sobre la tabla, para la cual se realizará una configuración de réplica, el sistema capturará la acción con los valores de los cambios realizados. Se crea un objeto de tipo grupo replicable (ReplicableGroup), el cual estará compuesto por una lista de acciones replicables (ReplicableAction). Para cada acción replicable se verificará si la misma contiene una columna referencia; en caso afirmativo, se debe obtener el fichero referenciado. Después de obtenido el fichero, el mismo es segmentado en dependencia del tamaño especificado en la configuración general del sistema. Se crea una lista de objetos de tipo PackageSend, con cada segmento obtenido del fichero. La lista obtenida es persistida en una BD embebida local. Luego de persistida la información, se procede a mapearla. Después de mapeada, se dará inicio al proceso de envío de fichero; se envía un objeto de tipo PackageSend hacia todos los nodos registrados. Para enviar el próximo paquete, se esperará la notificación de recibo por parte de los nodos receptores. Al enviar el último paquete, se generará una notificación de fichero completo. Al capturar la notificación de fichero reconstruido, se verificará si en todos los nodos destinos el fichero fue reconstruido satisfactoriamente; en caso afirmativo, se procederá a borrar la información almacenada en la BD embebida local y los objetos mapeados.	
Observaciones: NA	

Prototipo de interfaz: NA

Tabla 8 HU Reconstruir fichero

Número: HU 2	**Nombre del requisito:** Reconstruir fichero
Programador: Walter Jacas Jardines	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3 semanas
Riesgo en desarrollo: • Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas eléctricos.	**Tiempo real:** 3 semanas
Descripción: Al arribar al servidor de mensajería un objeto de tipo PackageSend, el sistema procesará la información contenida en el nodo receptor, se verifica si el paquete recibido es el esperado; en caso afirmativo, persistirá el objeto en la BD embebida local. Luego de haber almacenado la información, el sistema procederá al envío de una notificación de recibo al nodo emisor. En caso negativo, donde el paquete recibido no es el esperado, el sistema enviará una notificación al nodo emisor, en la misma se especifica el número del paquete esperado. Al recibir la notificación de fichero completo y almacenar el último paquete, el sistema procesará la información almacenada, procediendo a la reconstrucción del fichero.	
Observaciones: NA	
Prototipo de interfaz: NA	

2.7 Arquitectura del software REKO

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés), en su norma número 1471-2000, define a la arquitectura de software como: “La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos así como el ambiente y los principios que orientan su diseño y evolución.” (46)

Un estilo arquitectónico se define como: “una lista tipos de componentes que describen los patrones o las interacciones a través de ellos. Un estilo afecta a toda la arquitectura de software y puede combinarse en la propuesta de solución”. (47)

El desarrollo Basado en Componentes es un estilo arquitectónico utilizado en aplicaciones que pueden descomponerse en componentes lógicos o funcionales. A su vez, los componente de software son unidades reutilizables que pueden relacionarse con otros módulos de software por medio de interfaces bien definidas que contienen métodos, eventos y propiedades, que pueden estar escritos en lenguaje funcional, procedural u orientado a objetos. (48)

En la especificación UML se define que: "…un componente es una unidad modular con interfaces bien definidas, que es reemplazable dentro del contexto". (47)

Este estilo arquitectónico permite simplificar las pruebas al facilitar realizarlas en cada uno de los componentes por separado antes de probar el conjunto completo de componentes ensamblados. Amplía la calidad del software debido a que un componente puede ser perfeccionado por los desarrolladores sin afectar el resto de los componentes y simplifica el mantenimiento del sistema, entre otras cuestiones. Es utilizado para extender el nivel de productividad de los grupos desarrolladores así como optimizar la fiabilidad, flexibilidad y la reutilización de la aplicación final. (49)

Teniendo como base las definiciones antes expuestas, se puede definir en términos generales la arquitectura de REKO, como basada en componentes, pues sus partes encapsulan un conjunto de comportamientos que pueden ser reemplazados por otros. Como se muestra en la figura 5 se pueden observar los componentes de REKO:

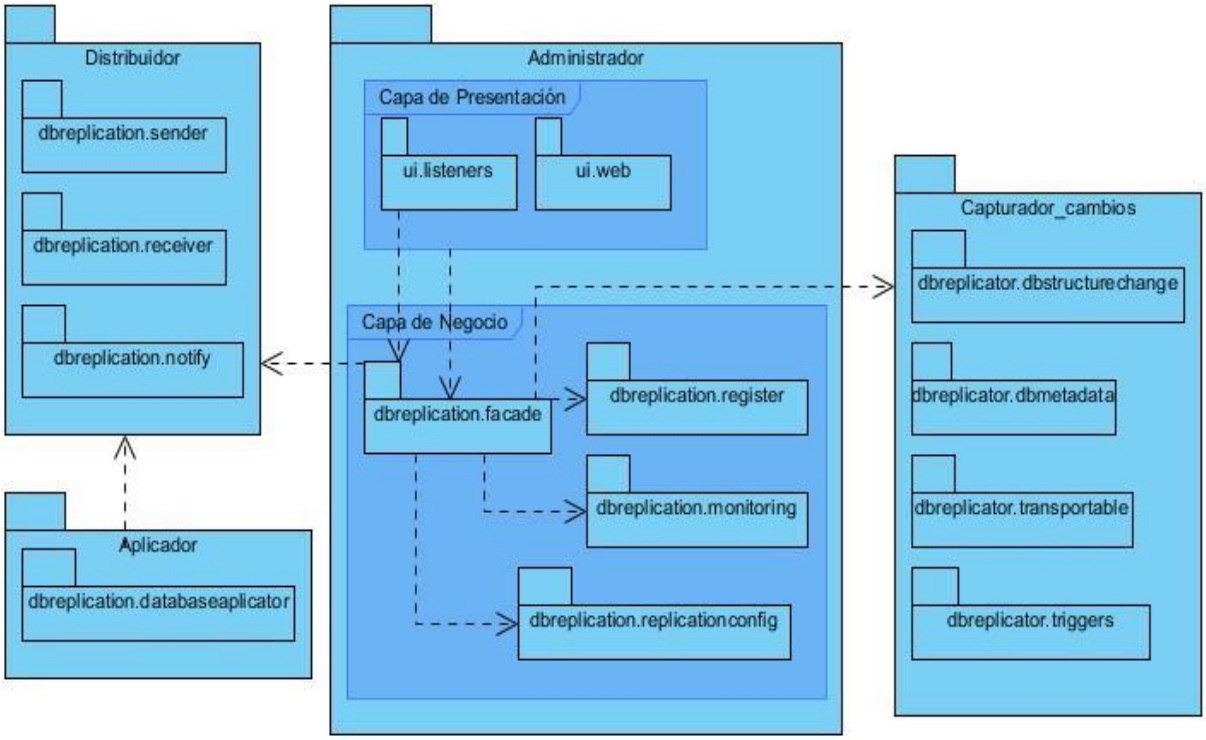


Figura 4 Componentes de REKO

Arquitectura en capas

Independientemente de lo antes expuesto, el componente Administración, como puede observarse en la figura 5, responde a un modelo multicapas, donde cada capa tiene funcionalidades y objetivos precisos. Así su implementación se encuentra desacoplada de la programación de cualquier otra y la comunicación con una capa inferior ocurre a través de interfaces. Además de estar separadas lógica y estructuralmente, las capas se encuentran separadas de manera física.

La **capa de presentación** es la que el usuario ve en su ordenador, es donde se tratan los datos que se van a mostrar. Esta capa se comunica únicamente con la capa de negocio.

La **capa de negocio** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina además como lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. (46)

Para la implementación del componente, que llevara a cabo el proceso de réplica de ficheros, se hará uso de los componentes Capturador de Cambios, Distribuidor de Cambios, Administración y Aplicador de Cambios.

2.8 Patrones de diseño

Los patrones de diseño pueden ser definidos como la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño; facilitan la reusabilidad, extensibilidad y mantenimiento. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (50; 53)

A continuación se describen los patrones de diseño empleados:

Patrón Objeto de Acceso a Datos

Independiza la aplicación de la forma de acceder a la base de datos, o cualquier otro tipo de repositorio (base de datos, archivos, servicios externos, etc.). Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos, o de cuál es la fuente de almacenamiento. Para ello tiene un interfaz común, sea cual sea el modo y fuente de acceso a datos, así como la clase Objeto de Acceso a Datos (DAO, por sus siglas en inglés), que centraliza el código relativo al acceso al repositorio de datos (ver Anexo 27). Fuera de las clases DAO no debe haber ningún tipo de código que acceda al repositorio de datos. El DAO accede a la fuente de datos y la encapsula para los objetos clientes. Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente (API de Persistencia Java), la cual podría ser JDBC.(52)

Los patrones Generales de Software para Asignar Responsabilidades *(*GRASP*,* por sus siglas en inglés*),* describen los principios fundamentales de asignación de responsabilidades a objetos. (53)

Experto: asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Aplica la idea de que los objetos realizan acciones relacionadas con la información que posee. Por ejemplo: La clase *SenderManager* cuenta con la información necesaria para cumplir la responsabilidad del envío de ficheros, mientras que la ReceiverManager se responsabiliza de recepcionar el fichero enviado.

Creador: cada instancia de un objeto es creada por el objeto que tiene la información necesaria para ello. Por ejemplo, a la clase FileUtil se le asigna la responsabilidad de crear una instancia de *PackageSend* porque:

- FileUtil utiliza específicamente los objetos PackageSend.
- FileUtil tiene los datos de inicialización que serán transmitidos a PackageSend cuando este objeto sea creado.

Por tanto *FileUtil* es un creador de los objetos de PackageSend.

Controlador: es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema. Un Controlador define el método para la operación del sistema.

Las clases controladoras diseñadas SenderManager, ReceiverManager, AplicatorManager y NotifyManager, controlan cada flujo de eventos del sistema, lo que facilita la centralización de actividades.

Alta Cohesión: como cada clase tiene un conjunto de funcionalidades relacionadas directamente con la entidad que representan, no realizan un trabajo enorme y por tanto, pueden ser calificadas como de alta cohesión. La clase JmsSenderFile tiene una responsabilidad moderada en el área de envío de ficheros, y colabora con otras clases para llevar a cabo las tareas.

Bajo acoplamiento: las clases se comunican solo con las clases necesarias para desarrollar cada flujo de evento. Por ejemplo para el flujo del envío de fichero existe un bajo acoplamiento (ver Anexo 28).

Patrones Banda de Cuatro (GOF, por sus siglas en inglés)

Fachada (*Facade*): provee de una interfaz unificada simple, para acceder a una interfaz o grupo de interfaces de un subsistema. Se ha utilizado este patrón para reducir la dependencia entre clases, ofrece un punto de acceso, de manera que si estas cambian o se sustituyen por otras, solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. REKO tiene el componente facade que aprovisiona estas ventajas. (54)

2.9 Modelo de diseño

2.9.1 Diagrama de paquetes del diseño

Los paquetes nos permiten describir la arquitectura de un sistema con la notación UML. Se incluye una vista alterna, que representa los agrupamientos lógicos mediante los diagramas de paquete de UML. A este tipo de diagrama podemos llamarlo diagrama de paquetes de la arquitectura. (53)

A continuación se muestran en las figuras 5 y 6, los diagramas de paquetes de las principales acciones que debe realizar el componente a desarrollar.

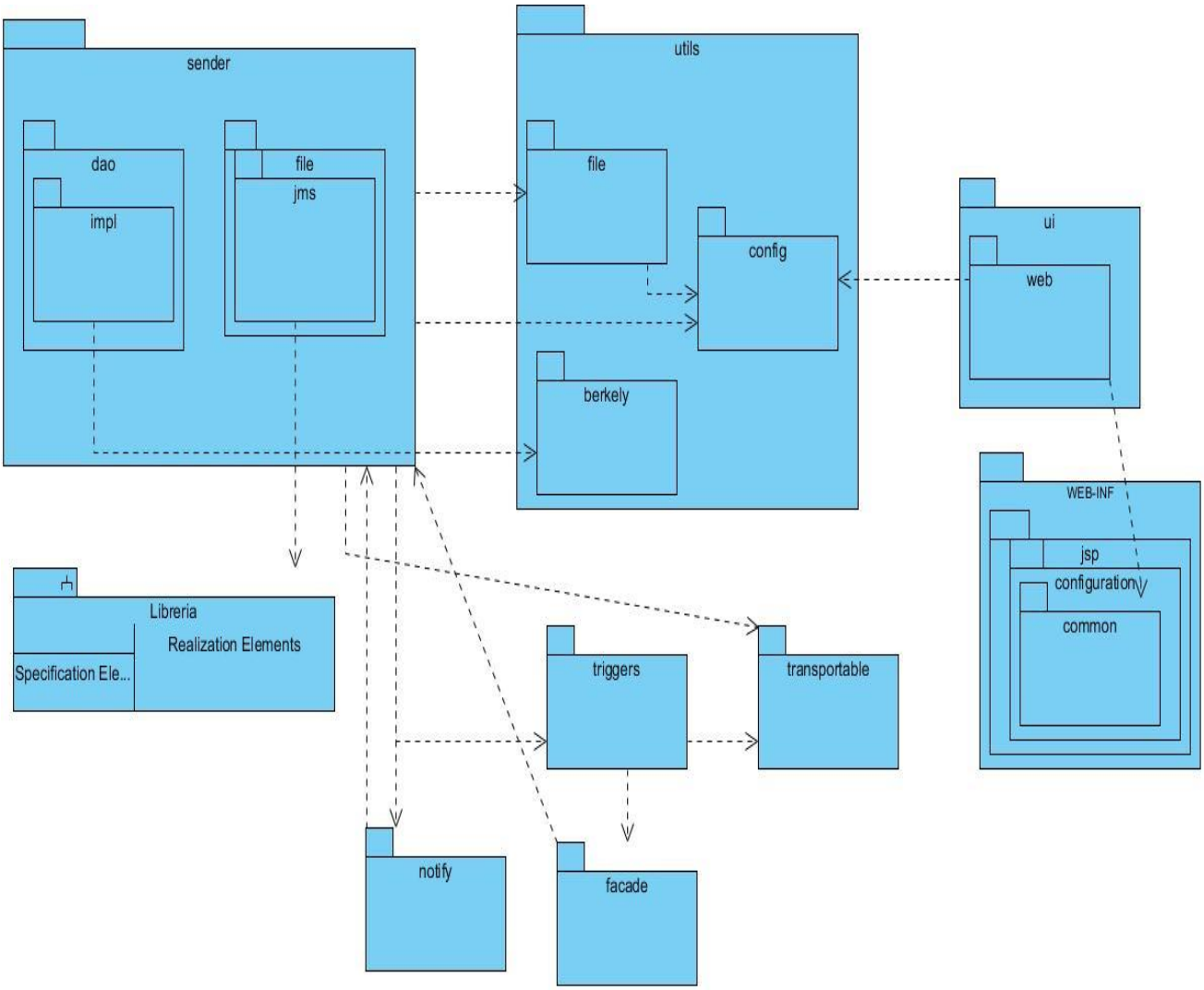


Figura 5 Diagrama de paquetes del requisito funcional enviar fichero

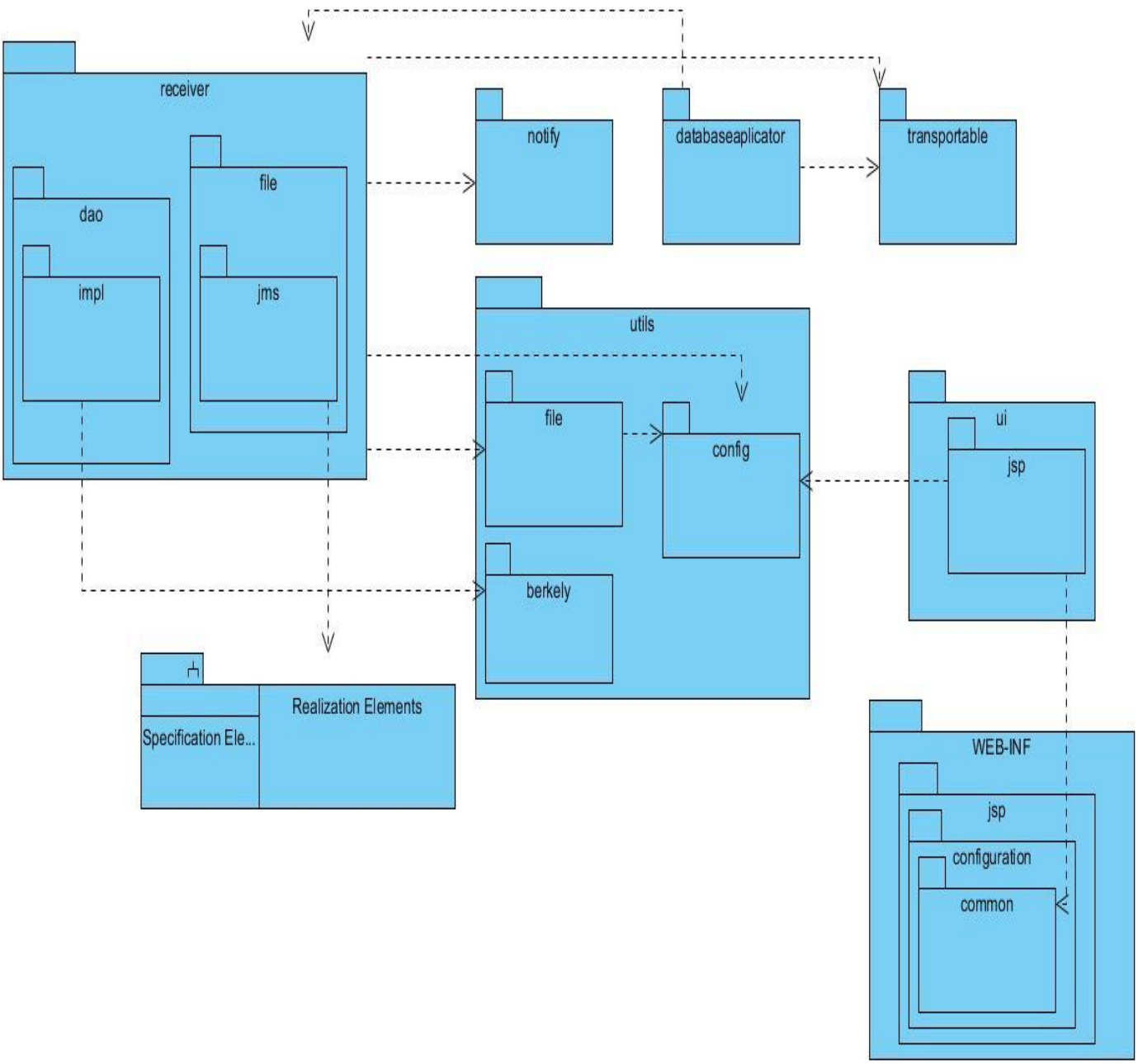
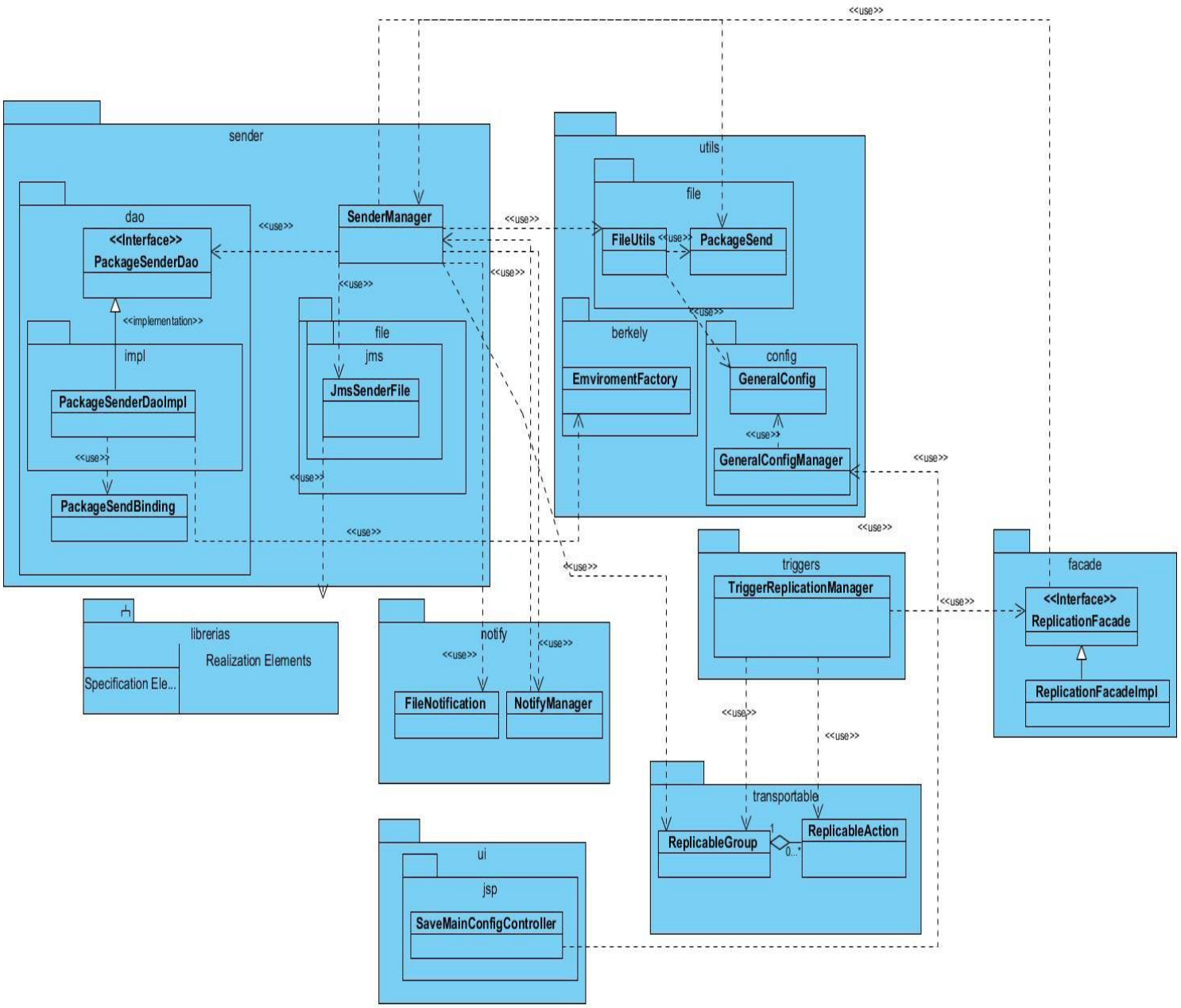


Figura 6 Diagrama de paquetes del requisito funcional construir fichero

2.9.2 Diagrama de clases del diseño

Los diagramas de clases son una representación concreta de lo que debe ser implementado. (14)

A continuación se muestran en las figuras 7 y 8 los diagramas de clases de las principales



acciones que debe realizar el componente a desarrollar.

Figura 7 Diagrama de clases del diseño para el envío de ficheros

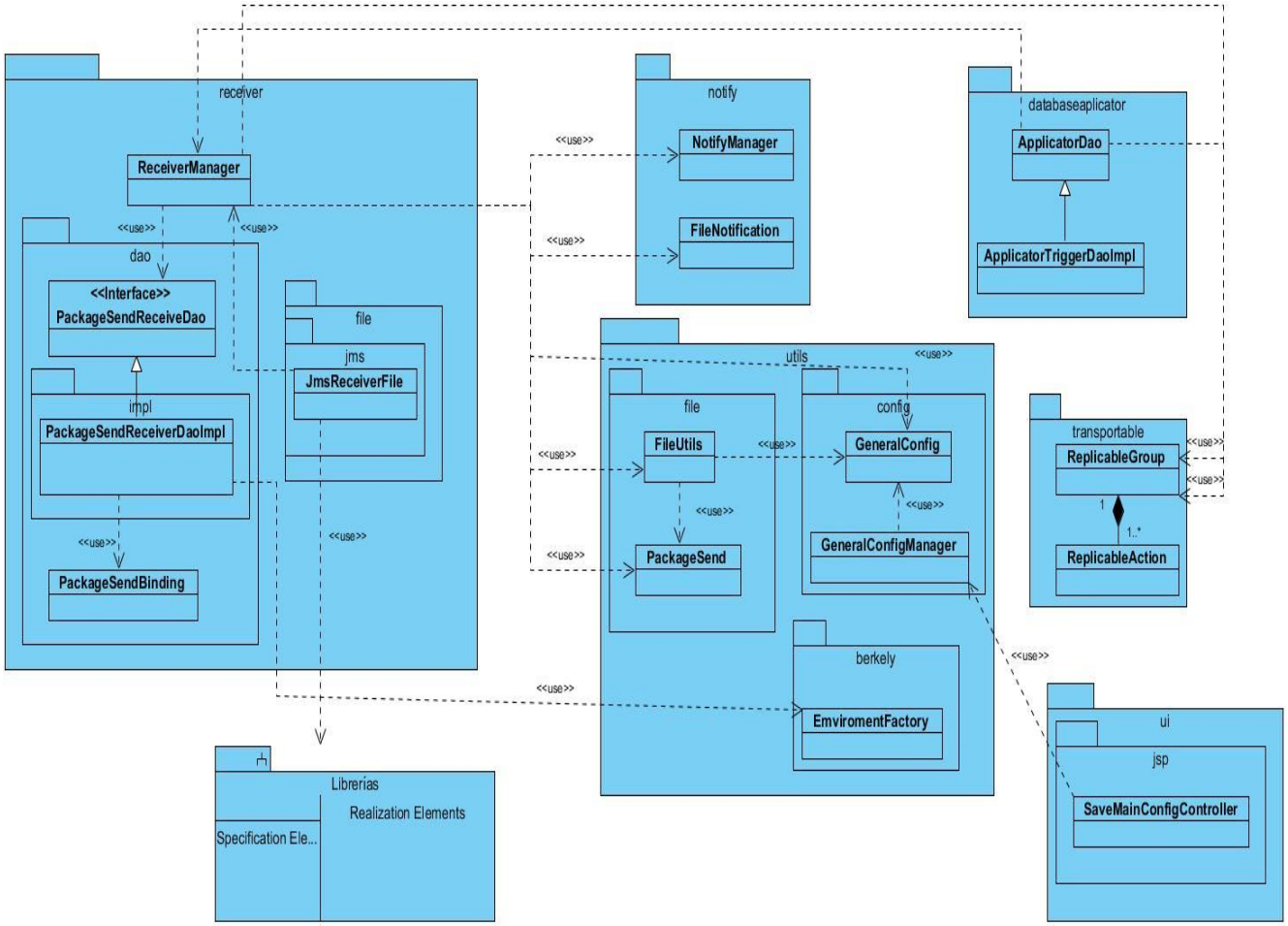


Figura 8 Diagrama de clases del diseño para la captura de fichero

2.9.3 Descripción de las clases

Tabla 9 Descripción de la clase ReceiverManager

Descripción de la clase ReceiverManager	
Nombre: ReceiverManager.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
fileCompleted	private
receivedDao	private
Responsabilidades	
Contiene todos los métodos get y set para obtener así como modificar el atributo especificado respectivamente.	

Nombre	onFileArrive(package: PackageSend)
Descripción	Captura objeto de tipo PackageSend enviado por nodo emisor
Nombre	onDeleteReference(url: String)
Descripción	Elimina fichero, recibe como parámetro la dirección del fichero que se desea eliminar
Nombre	cleanBDReceived(package: LinkedList<PackageSend>)
Descripción	Borra información almacenada en BD embebida, al reconstruir fichero.
Nombre	ControlPackageCorrectReceived(package: PackageSend)
Descripción	Verifica qué paquete recibido es el esperado.
Nombre	removeFileDirectoryByActionDelete(parametersAndActionType: List<ParametersAndActionType>)
Descripción	Elimina fichero si acción que se ejecuta es de eliminación.

2.10 Modelo de despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, además de las instancias de los componentes y objetos que residen en ellos. El Modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos.(42)

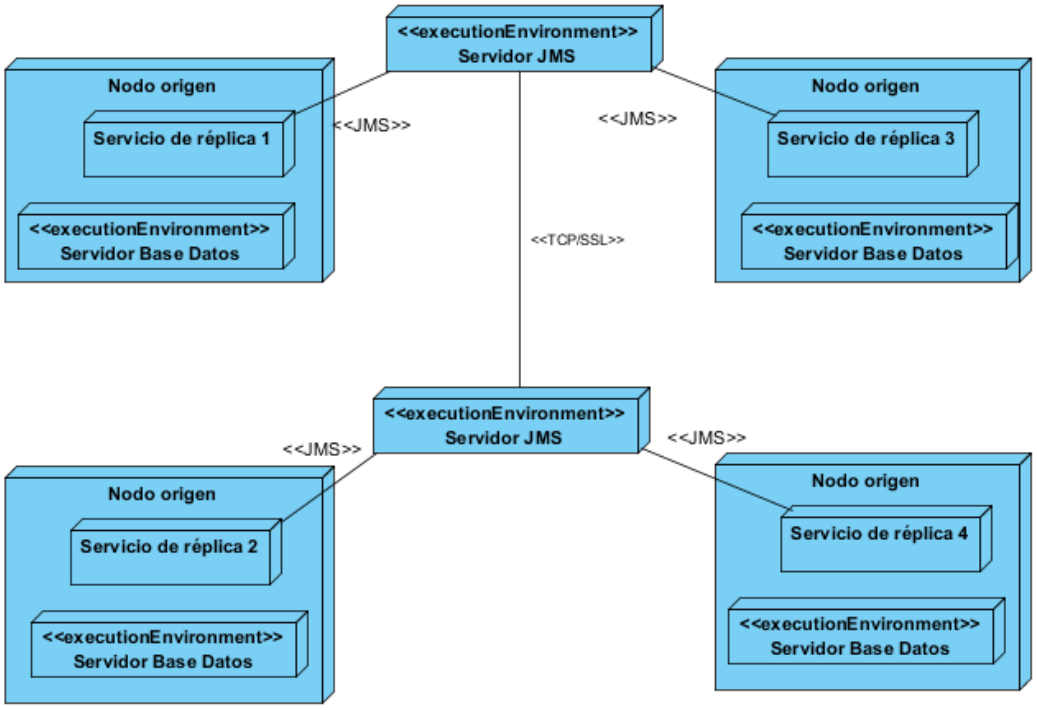


Figura 9 Diagrama de despliegue

El modelo de despliegue propuesto en la figura 9, es un ejemplo de un posible escenario de réplicas, el cual está compuesto por 4 nodos, donde en cada uno de ellos se tiene una instancia de REKO. El replicador lleva a cabo el proceso de monitoreo de las posibles operaciones que realicen en la BD, dígase inserción, actualización o eliminación. Los nodos propuestos están conectados entre sí a través de dos servidores JMS, para llevar a cabo el proceso de envío de información entre nodos.

2.11 Consideraciones parciales del capítulo

Teniendo en cuenta lo antes planteado se arriban a las siguientes consideraciones parciales:

- Se realizó el Modelo de Dominio del Negocio, pues no existen procesos observables; siendo definidas como clases cada una de las partes.
- A partir de la descripción del proceso a automatizar se logró la identificación de requisitos, tanto funcionales como no funcionales.
- Se hace uso de las historias de usuarios, para la descripción de los requisitos funcionales definidos.
- El componente está integrado por dos partes fundamentales: una encargada del envío de ficheros y otra del recibo de los mismos; esta última contiene las acciones necesarias para llevar a cabo el proceso de reconstrucción de ficheros en el nodo destino.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS DEL COMPONENTE PARA LA RÉPLICA DE FICHEROS EN EL REPLICADOR DE DATOS REKO

En el presente capítulo se abordan contenidos relacionados con la descripción de la implementación del componente para la réplica de ficheros en REKO, con el fin de darle solución a las historias de usuario definidas en el capítulo anterior. Además, se abordarán detalles del diseño, la ejecución y los resultados de las pruebas que le serán aplicadas al sistema.

3.1 Tareas del programador

Para darle solución a las HU definidas en el capítulo anterior, se definen para cada una de ellas un conjunto de tareas a desarrollar, que se pueden observar en la tabla 10.

Tabla 10 Tareas del programador

Historia de usuario	Tarea
Enviar fichero	• Desarrollo de una entidad PackageSend, la cual contendrá la información necesaria del fichero que se analice. • Desarrollo de una funcionalidad que lleve a cabo el proceso de particionamiento de fichero, en dependencia del tamaño de buffer definido en la configuración general de REKO. • Desarrollo de una funcionalidad que permita persistir objetos de tipo PackageSend, en una BD embebida. • Desarrollo de una funcionalidad para llevar a cabo el proceso de obtención y envío de fichero, hacia todos los nodos destinos. • Desarrollo de una funcionalidad para el envío y recepción de notificaciones.

Reconstruir fichero	• Desarrollo de una funcionalidad que permita persistir objetos de tipo PackageSend en una BD embebida, en el nodo destino. • Desarrollo de una funcionalidad que permita obtener el objeto de tipo PackageSend, enviado desde el nodo emisor. • Desarrollo de una funcionalidad para el envío y recibo de notificaciones. • Desarrollo de una funcionalidad para llevar a cabo el proceso de reconstrucción de fichero en el nodo destino.

Condiciones para el funcionamiento del componente

Para el correcto funcionamiento de la propuesta que se realiza en este informe de investigación, se requiere garantizar una adecuada configuración de réplicas Se emplea la opción de réplica de ficheros, disponible en la configuración general del software REKO.

3.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del Modelo de diseño, se implementan en términos de componentes, ficheros de código fuente, ejecutables, etc. (44)

3.2.1 Estándares de codificación

Los estándares de codificación son métodos empleados por el desarrollador, para lograr entender de manera rápida, fácil y sencilla, el código empleado en el desarrollo de un software. Además, garantizan un mantenimiento óptimo de dicho código por parte del programador además de mejorar la legibilidad, lo que permite que otras personas puedan colaborar, ya que la redacción del código no les resulta incómoda.

A continuación se muestra el estándar de codificación que se llevó a cabo en la implementación del componente propuesto.

Declaración de variables: las variables declaradas deben comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser representada con letra mayúscula.

```
private PackageSenderDao berkeleyPackageSenderDao;

private ControlNumberFileBuildDao berkeleyNumberFileDao;
```

Declaración de funciones: no debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este con la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra, debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

```
public void sendFileSplitted(ReplicableAction action, String transactionId, int actionIndex)
throws Exception {

public void onFileArrive(PackageSend paquete) {
```

Nombre de las clases: las clases comenzarán con mayúscula, si el nombre de la clase es compuesto, empezarán con mayúsculas las dos letras iniciales de cada palabra.

```
public class JmsSenderFile implements InitializingBean {
```

Se emplea el estándar de desarrollo empleado en REKO, los nombre de las variables, funciones y clases fueron puesto en inglés.

3.2.2 Diagrama de componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente, o ejecutables. Modelan la vista estática de un sistema. (44)

Como REKO presenta una arquitectura basada en componentes, se emplearon los subsistemas de implementación para realizar los diagramas de implementación, dividiéndolo en partes más pequeñas, que puedan ser integradas y probadas de forma separada.

Los subsistemas organizan la vista de realización de un sistema. Son un agrupamiento semántico de útil de clases o de otros subsistemas que posee un grupo de interfaces. (46) En las figuras 10 y 11, se representan los diagramas de componentes de las principales acciones a desarrollar en el componente para la réplica de ficheros, que son las acciones de envío de ficheros, seguido por la de reconstrucción de ficheros.

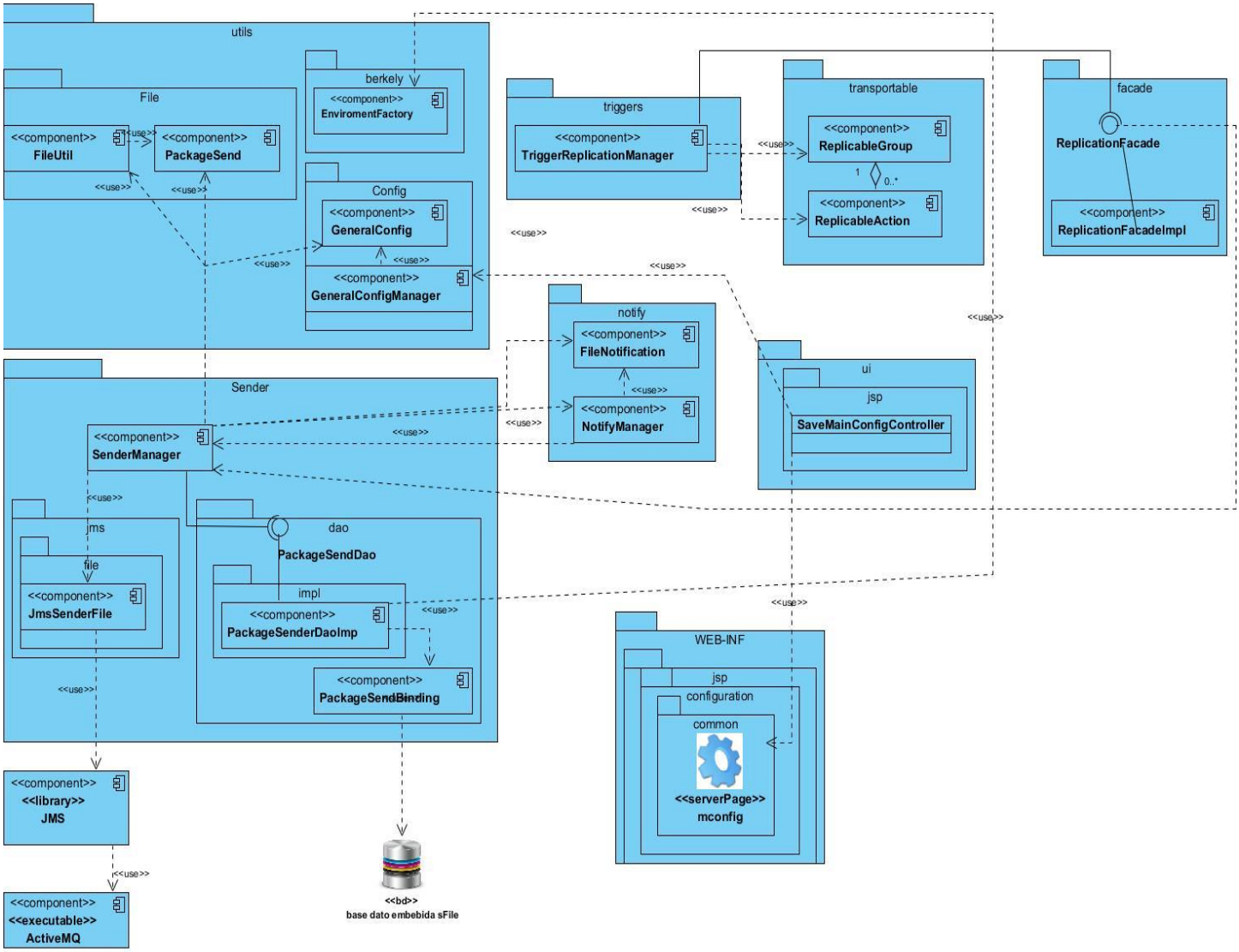


Figura 10 Diagrama de componentes enviar fichero

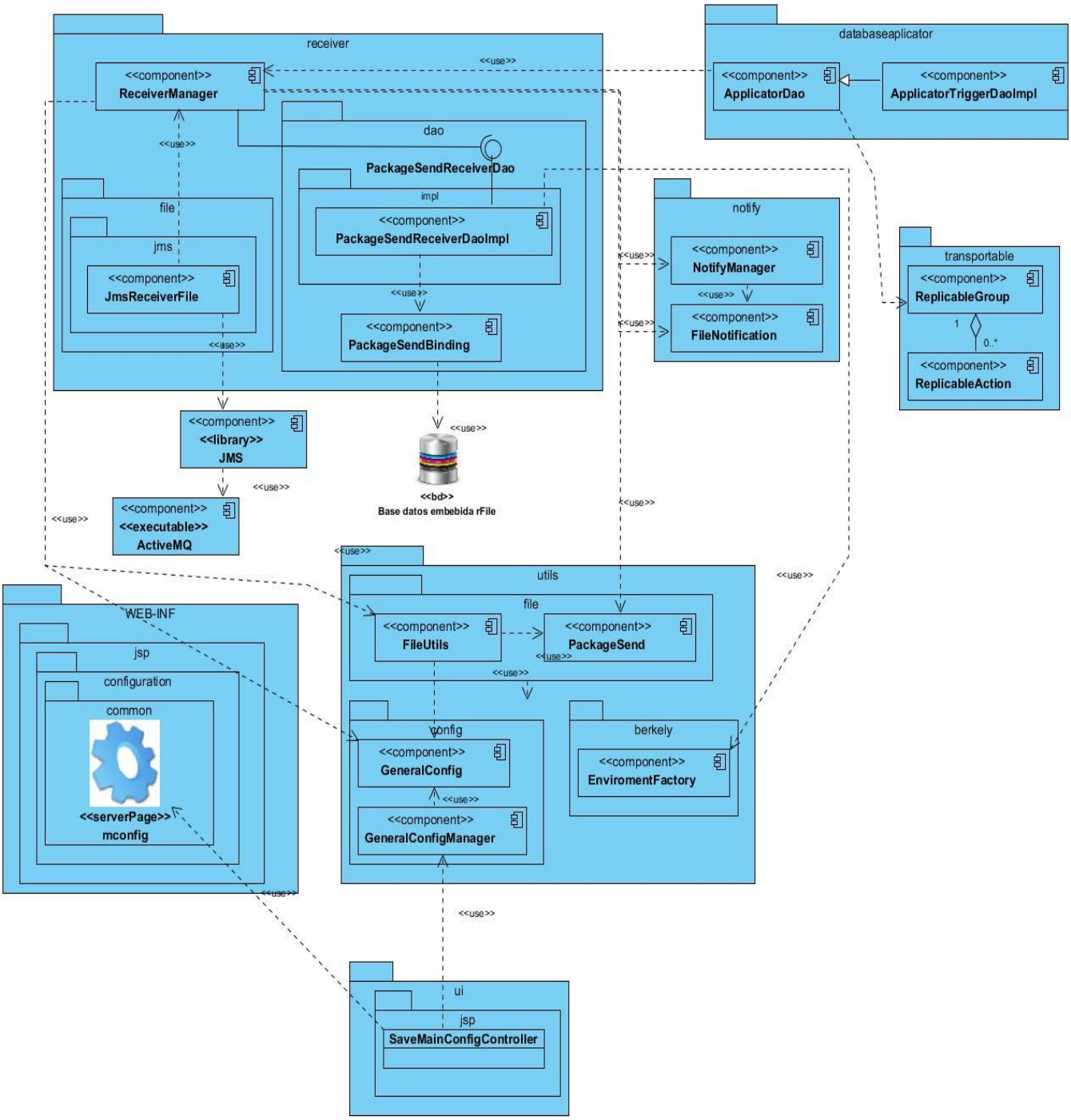


Figura 11 Diagrama componentes recepción de fichero

3.3 Modelo de pruebas

Se puede definir que: “las pruebas de un software, constituyen una actividad en la cual, un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente”. (56)

3.3.1 Técnicas y estrategias de prueba

Existen dos técnicas para realizar pruebas de validación, las técnicas de caja blanca o estructural, que se basan en un riguroso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Se aplican además las técnicas de caja negra o funcional, donde se realizan pruebas sobre la interfaz del programa a probar. Se entiende por interfaz las entradas y salidas de dicho programa, no es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (57)

Además, para evaluar dinámicamente un sistema de software se definen estrategias o niveles de prueba que permiten probar desde los componentes más simples y pequeños, e ir avanzando hasta probar todo el software en su conjunto. En total, se dividen en cinco niveles de pruebas.

- Pruebas unitarias: son la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, esté correctamente codificado. En estas pruebas, cada módulo será probado por separado y lo hará, generalmente, la persona que lo creó.
- Pruebas de integración: son realizadas para probar el software ensamblando todos los módulos probados previamente, a partir del esquema del diseño.
- Pruebas del Sistema: son efectuadas cuando el software se encuentra ensamblado totalmente e integrado con cualquier componente hardware. Permite comprobar que se cumplen los requisitos funcionales.
- Pruebas de aceptación: son aplicadas cuando el producto está listo para implantarse en el entorno del cliente y son realizadas por el usuario en conjunto con las personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de prueba.
- Pruebas de regresión: consisten en la repetición selectiva de pruebas para detectar fallos introducidos durante la modificación de un sistema o componente

de un sistema, y son efectuadas para comprobar que los cambios no han originado efectos adversos no intencionados. (57)

Al componente desarrollado se le aplicaron las pruebas de aceptación y las pruebas unitarias. En la tabla 14, se listan las pruebas aplicadas al componente desarrollado.

Tabla 11 Pruebas realizadas al componente

No	Caso de prueba	Resultado esperado
1.	Configuración de los parámetros para la réplica de ficheros en la configuración general de REKO.	El sistema guarda la nueva configuración realizada en el módulo de configuración general.
2.	Reconstrucción de fichero y comparación con fichero origen.	Mostrar mensaje en consola de reconstrucción satisfactoria del fichero.
3.	Persistencia de datos en BD embebida a partir de un objeto de tipo File.	Mostrar en consola mensaje de persistencia exitosa.
4.	Obtener valores almacenados en BD embebida.	Lista de objetos de tipo PackageSend.
5.	Proceso de réplica por la operación de INSERT con columna referencia en los gestores de BD PostgresSQL, ORACLE, MYSQL y SQLSERVER.	Reconstrucción satisfactoria del fichero en todos los nodos destinos.
6.	Proceso de réplica por la Operación UPDATE con columna referencia en gestores de BD PostgresSQL, ORACLE, MYSQL y SQLSERVER.	Reconstrucción satisfactoria del fichero en todos los nodos destinos.
7.	Proceso de réplica por la Operación DELETE con columna referencia en los gestores de BD PostgresSQL, ORACLE, MYSQL y SQLSERVER.	Reconstrucción satisfactoria del fichero en todos los nodos destinos.

3.3.2 Validación de las variables empleadas en la investigación

Las variables definidas en la investigación se listan y describen grosso modo a continuación:

- Fiabilidad.
- Calidad.
- Consistencia.

El autor de la presente investigación, se afilia al criterio de que la fiabilidad es el buen funcionamiento de algo, que ofrece seguridad o buenos resultados. El principal criterio a evaluar, para validar la fiabilidad, es que el componente a desarrollar deberá garantizar que en todas las instancias de REKO, presentes en el escenario de réplica, el fichero será reconstruido satisfactoriamente.

En este informe, se define por calidad la propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. El componente para la réplica de ficheros, deberá llevar a cabo el proceso de replicación de fichero satisfactoriamente, sin necesidad de hacer uso del servicio FTP.

En correspondencia con la Real Academia de la Lengua Española se define por consistencia, la coherencia de los elementos de un conjunto. El componente que llevará a cabo el proceso de réplica de ficheros en REKO, deberá afirmar la coherencia entre los paquetes enviados, y garantizar que ninguno de ellos cuente con información corrupta. El hecho que ninguno de ellos cuente con información corrupta asegurará la reconstrucción satisfactoria del fichero en el nodo destino, lo que permite que coincida con el fichero enviado desde el nodo emisor.

Fueron ejecutados un total de cuatro escenarios de réplicas reales, para llevar a cabo el proceso de réplica de ficheros. Se hizo uso de los cuatro gestores de BD para los que REKO, brinda el servicio de réplica de datos. En cada escenario la cantidad de nodos registrados fue diferente; se transmitió la información desde un nodo maestro hacia los nodos esclavos, con el objetivo de validar las variables definidas en la investigación, arrojando los siguientes resultados:

- El proceso de réplica de ficheros fue ejecutado, sin el uso de un servicio FTP.

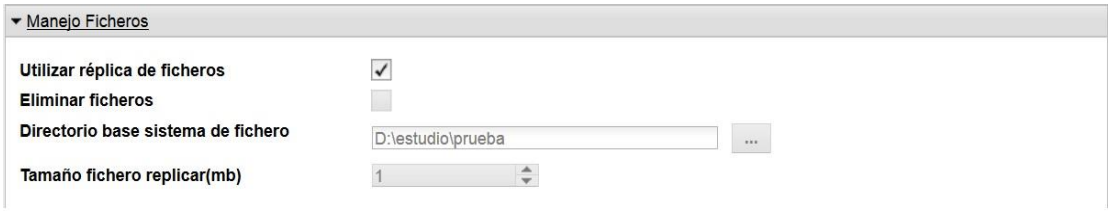
- En cada nodo destino, fue reconstruido satisfactoriamente el fichero replicado.
- El fichero reconstruido en cada uno de los nodos destinos, coincidió con el fichero enviado desde el nodo emisor.

3.3.3 Prueba de aceptación

Tabla 12 Configuración de los parámetros para la réplica de fichero

Código: Caso de prueba 1
Nombre: Configurar los parámetros para la réplica de ficheros.
Descripción: Configuración de los nuevos parámetros necesarios para llevar a cabo el proceso de réplica de ficheros.
Condiciones de ejecución. • El usuario debe estar autenticado en el sistema. • Usuario autenticado debe tener permiso al módulo para el manejo de ficheros.
Entrada/Pasos de Ejecución: • **Réplica de ficheros:** true. • **Dirección directorio base de ficheros:** D:\estudio\prueba. • **Eliminar fichero del directorio si operación ejecutada en la BD es DELETE:** true • **Tamaño del buffer de fichero:** 1 MB.
Resultado esperado: Se almacenará los nuevos parámetros entrados en la configuración general del sistema, para llevar a cabo el proceso de réplica de ficheros.

Resultado obtenido:



Evaluación de la prueba. Satisfactoria

3.3.4 Pruebas unitarias

Actualmente existen varias herramientas que facilitan la elaboración y desarrollo de los casos de pruebas, además de darle seguimiento a los errores. Un ejemplo de ello es JUnit, utilizada para realizar las pruebas unitarias en aplicaciones Java.

Para la ejecución de estas pruebas fue necesario diseñar una clase denominada FileUtilsTest, la cual contenía los casos de pruebas a realizar en correspondencia con los métodos de la clase FileUtils para el manejo de ficheros. Fueron replicados seis ficheros con extensión .doc, cuatro con extensión .pdf, uno con extensión .txt, dos archivo con extensión .mp3, y por último dos archivos con extensión .avi para un total de 15 archivos replicados. Desde la figura 12 hasta la figura 17 se muestran los resultados en las dos iteraciones realizadas:

Figura 12 Prueba unitaria para validar la variable Calidad. Primera iteración

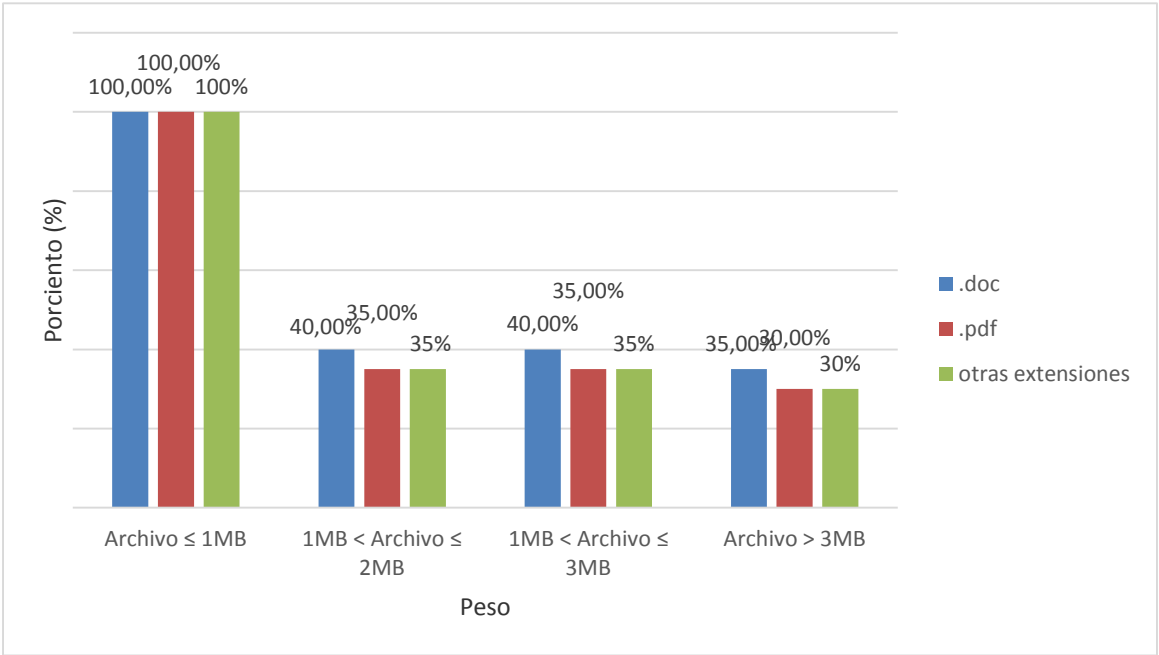


Figura 13 Prueba unitaria para validar la variable Fiabilidad. Primera iteración

Figura 14 Prueba unitaria para validar la variable Consistencia. Primera iteración

En la primera iteración de las pruebas realizadas, se detecta un error de complejidad media en el proceso de reconstrucción de ficheros en los nodos destinos, cuyo tamaño era superior a 1MB. De los cinco ficheros replicados con extensión .doc, donde su tamaño fue superior a 1MB, solo dos pudieron reconstruirse satisfactoriamente. Se replicaron tres archivos .pdf con tamaño superior a 1MB, de ellos solo uno pudo reconstruirse correctamente. De los 5 archivos con otras extensiones, tres de ellos, superaban 1MB de tamaño, de ellos solo uno pudo reconstruirse convenientemente. El error detectado fue solucionado, arrojando los siguientes resultados en la segunda iteración de las pruebas unitarias realizadas.

Figura 15 Prueba unitaria para validar la variable Fiabilidad. Segunda iteración

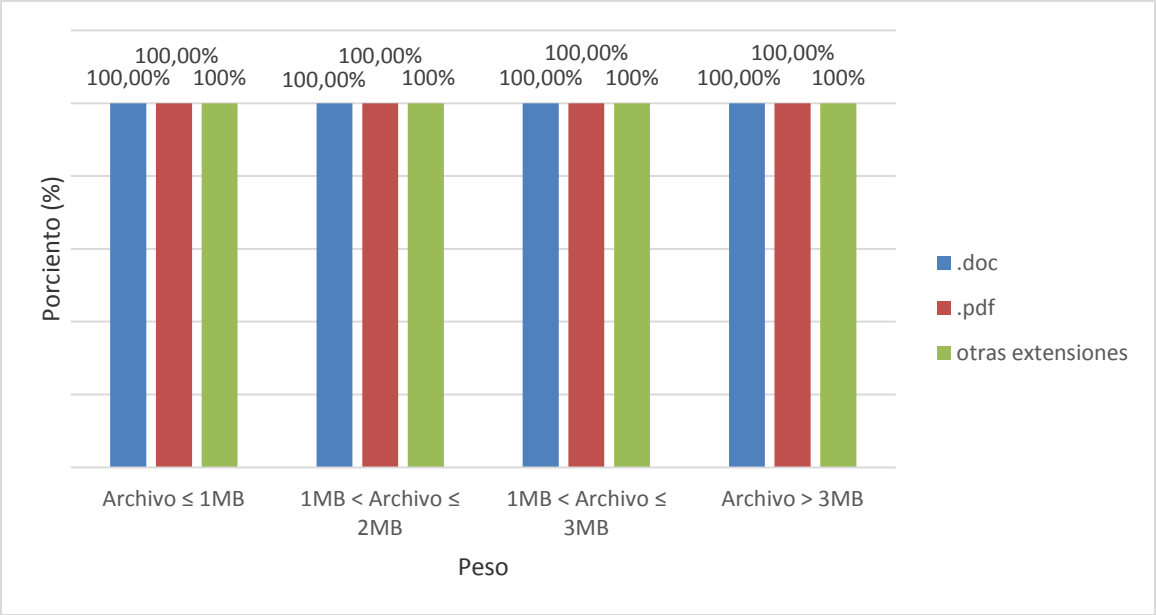


Figura 16 Prueba unitaria para validar la variable Calidad. Segunda iteración

Figura 17 Prueba unitaria para validar la variable Consistencia. Segunda iteración

3.3.5 Resumen de las pruebas realizadas

En el proceso de pruebas efectuado se diseñaron un total de 16 casos de pruebas, con el objetivo de verificar el correcto funcionamiento del componente. Se verificó el funcionamiento de la réplica de ficheros, con el que se garantizó la fiabilidad, calidad y consistencia de los datos, en un escenario de réplicas para el software REKO.

3.3.6 Resultados de las pruebas

Al finalizar la disciplina de pruebas se obtuvieron los siguientes resultados:

- El componente mostró ser completamente funcional, dando solución a todos los requisitos funcionales planteados.
- Durante el desarrollo del componente para la réplica de ficheros, fueron solucionados todos los problemas detectados en el proceso de réplica de ficheros.
- Todas las pruebas dieron resultados satisfactorios. De esta manera se demuestra que el componente cumple con cada una de las historias de usuarios que lo conforman.

3.4 Consideraciones parciales del capítulo

Teniendo en cuenta los aspectos anteriormente planteados se pueden arribar a las siguientes conclusiones parciales:

- Se cuenta con todas las funcionalidades necesarias para darle solución a cada requisito funcional definido en la investigación.
- La realización de las pruebas de aceptación y las unitarias, permitieron detectar, así como corregir las no conformidades existentes en el sistema implementado.
- El empleo de estas pruebas, permitió verificar la solidez de la implementación del componente para la réplica de ficheros en el Replicador de datos REKO.

CONCLUSIONES

Como fruto del decurso investigativo, se han arribado a las siguientes conclusiones:

1. En el estudio se realizó un análisis de los conceptos fundamentales y la formalización del estado del arte de los replicadores de datos, haciendo énfasis en el proceso de réplica de ficheros, evidenciándose la inexistencia de una variante concreta que le brinde solución a la situación problemática planteada en el diseño de la investigación.

2. Fueron descritas las características de la solución propuesta, definidas a partir de la modelación de los procesos de negocio y la especificación de los requisitos de software. Elementos que permitieron un entendimiento del problema, para ejecutar la implementación del componente para la réplica de ficheros.

3. Se desarrolló un componente capaz de llevar a cabo el proceso de réplica de ficheros en REKO, con el que se garantiza la fiabilidad, calidad y consistencia de los datos.

4. Al componente desarrollado se le aplicaron las pruebas de caja negra y de caja blanca, con el objetivo de validar la fiabilidad, calidad, así como la consistencia de los datos, en el proceso de réplica de ficheros. Finalizada las pruebas se pudo lograr el correcto funcionamiento de la solución propuesta en uno cada de los escenario de réplica probado.

RECOMENDACIONES

- Agregarle al componente desarrollado, una funcionalidad capaz de llevar a cabo el proceso de réplica de ficheros en modo sin conexión.
- Agregarle al componente desarrollado una funcionalidad capaz de llevar a cabo el proceso de réplica de ficheros en un escenario, donde los nodos tengan instalados sistemas operativos diferentes.

REFERENCIAS BIBLIOGRÁFICAS

1. COLECTIVO DE AUTORES. Elementos de informática básica. La Habana: Editorial Pueblo y Educación, 2000. 250 p. ISBN 959-13-0423-4.

2. DATE, C. J. V., JOSÉ LUIS (ED. LIT.). Introducción a los sistemas de bases de datos. Editado por: Ruiz Faudón, S. L. T. L. G., Felipe(Rev. Tec.). 7 ed. México: Pearson Educación, 2001. 960 p. ISBN 968-444-419-2.

3. SILBERSCHATZ, A. Fundamentos de bases de datos Editado por: Vázquez, J. L. 4 ed. España: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2002. 787 p. ISBN 0-07-228363-7

4. SIERRA, M. ¿Qué es una Base de Datos y cuáles son sus principales tipos? ,2013 .[en línea],. Disponible en: http://www.aprenderaprogramar.com/index.php?option=com_attachments&task=download&id=500.

5. INSTITUTO TECNOLÓGICO DE VERACRUZ. Creación del esquema de la base de datos.[en línea]. Disponible en: http://www.prograweb.com.mx/tallerBD/0201Esquema.html.

6. CARRANZA ATHÓ, F. Bases de datos distribuidas. Perú: Escuela Académico Profesional de Informática, 2006.

7. URBANO, R. Oracle® Database Advanced Replication 11g Release 2 (11.2) Oracle®, Octubre 2010 .[en línea]. Disponible en: http://docs.oracle.com/cd/E11882_01/server.112/e10706.pdf.

8. CRISTIAN MAZILU, M. Database Replication. Bucharest, Romania: Academy of Economic Studies, 2010. vol. I, no. 2/2010, 6 p.

9. SQL SERVER 2012. Replicación de SQL Server .[en línea]. 2012. Disponible en: http://msdn.microsoft.com/es-es/library/ms151198.aspx.

10. Manual de SQL. [en línea]. Disponible en: http://www.webexperto.com.

11. AOYAMA, M. New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development ? Japón: Niigata Institute of Technology, 1998.

12. SZYPERSKI, C. Component software: Beyond object-oriented programming .[en línea]. Disponible en: http://books.google.es/books/about/Component_software.html?hl=es&id=ZKlQAAAAMAAJ.

13. CRNKOVIC, I. L., MAGNUS. Building Reliable Component-Based Software Systems. Editado por: Crnkovic, I. L., Magnus. Boston: Artech House 2002. 492 p. ISBN 1-58053-327-2.

14. SOMMERVILLE, I. Ingeniería del Software. Editado por: Alfonso Galipienso, M. I. B. M., A.; Mora Lisán, F.; Trigueros Jover, J. P. (Trad.). 7ma ed. Madrid: Pearson Educación S.A. , 2005. 712 p. ISBN 84- 7829- 074- 5.

15. Real academia de la lengua Española. [En línea] Disponible en:

http://lema.rae.es/

16. Software daffodil. [En línea] Disponible en:

http://enterprise.replicator.daffodilsw.com/.

17. TUNGSTEN REPLICATOR. Guide, Tungsten version 2.0.4.[en línea]. Disponible en: http://s3.amazonaws.com/releases.continuent.com/doc/replicator-2.0.4/Tungsten-Replicator-Guide.pdf.

18. Tilma, B. J. BD Replicador. 2008e.

19. SYMMETRICDS. What is SymmetricDS ? .[en línea],. Disponible en: http://www.symmetricds.org/about/overview.

20. PIMENTEL GONZÁLEZ, L. A. H. S., EIVYS; BERMÚDEZ PEÑA,ROLANDO; PÉREZ ALFONSO,DAMIÁN; MENA RODRIGUEZ,LUIS EDGARDO; ALFONSO VALDÉS,JAVIER; GOMEZ PEÑA, ANGELA GLORIA Reko Replicador. Cuba: Universidad de las Ciencias Informáticas, 2009. 9 p.

21. COMPANIONI SARDIÑA, Y. A. R., ALBIN; HERNÁNDEZ SUAREZ,EYVIS; VELAZQUEZ VIZCAY,ADRIEL; NUÑEZ RIOS,MADIELENNIS; PÉREZ MATOS, LEISER Reko: una solución de réplica para sistemas de bases de datos relacionales distribuidos. Cuba: IV Simposio Informática y Comunidad, 2012. 9 p.

22. PRESSMAN, R. S. Ingenieria del Software, Un Enfoque Práctico. 6 ed. España: Editorial McGraw-Hil, 2002. 640 p. Técnicas para facilitar las especificaciones de la aplicación. ISBN 8448132149

23. **Torrecilla, Pablo.** [En línea]. Disponible http://nosolopau.com/2012/06/07/mas-sobre-el-proceso-unificado-agil-fases-y-disciplinas/.

24. **Sánchez, T.R.** Metodología de desarrollo para la actividad productiva de la UCI. La Habana: s.n., 2014.

25. VISUAL-PARADIGM. Febrero 2011 .[en línea],. Disponible en:

www.visual-paradigm.com.

26. GIRALDO, L. Z., YULIANA Herraminetas de desarrollo de ISW para Linux. 24 de Septiembre del 2005. 11 p. Disponible en: http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17.

27. ¿Qué es Java Enterprise Edition ? .[en línea]. Disponible en: http://www.java.com/es/download/faq/techinfo.xml.

28. WEITZENFELD, A. Ingeniería de software orientada a objetos con UML, Java e Internet. Editado por: Chavez Cabañas, R. México: Thomson Editores, S.A de S.V., 2005. [en línea]. 685 p. Disponible en:

http://books.google.com.cu/books?id=MOviEp0ApQcC&pg=PP6&lpg=PP6&dq=weitzenfeld+alfredo.+ingenier%C3%ADa+de+software+orientada+a+objetos&source=bl&ots=OWzHheQv9x&sig=PTmThWW1GaLbcSvn-

pvGzdqxJOY&hl=es&sa=X&ei=vykUUfS6NMnC0QGSzoCQAQ&ved=0CDMQ6AEwAQ. ISBN 970- 686 - 190 - 4.

29. HAMILTON, K. M., RUSSELL. Learning UML 2.0. O'Reilly, 2 de mayo del 2006.[en línea], .286 p. Disponible en:

http://eva.uci.cu/mod/resource/view.php?id=9352. ISBN 0-596-00982-8.

30. WALLS, C. Spring in Action. 3 ed. Estados Unidos: Manning Publications Co., 2008. 426 p. ISBN 9781935182351.

31. GUERRA, C. R. Configuración JMS en ORACLE WEBLOGIC v11.Lima,10 de mayo del 2012. [en línea]. Disponible en:

http://frameworksjava2008.blogspot.com/2012/05/configuracion-jms-en-oracle-weblogic.html.

32. THE APACHE SOFTWARE FUNDATION. Apache ActiveMQ .[en línea]. Disponible en: http://activemq.apache.org/index.html.

33. THE APACHE SOFTWARE FUNDATION. Apache Tomcat 9 de mayo del 2013. [en línea]. Disponible en: http://tomcat.apache.org/.

34. VOGEL, L. Java Unit testing with JUnit 4.x in Eclipse.Junio del 2007. [en línea]. Disponible en: http://www.eclipse.org/resources/resource.php?id=408.

35. PARI, J. Manual de Instalación de SpringSourse Tool Suite My JBreak. [en línea],. Disponible en: http://www.slideshare.net/juliopari/spring-tool-suite-instalacion.

36. GINESTA GILGERT, M. P. M., OSCAR. Bases de datos en PostgreSQL UOC, Disponible en: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02152.pdf.

37 SQL-Server [en línea], Disponible en:

http://www.microsoft.com/es-es/server-cloud/products/sql-server/

38. Oracle Database 12g, [en línea] Disponible en: http://www.oracle.com/technetwork/database/oracle-database-editions-wp-12c-1896124.pdf

39. BURBANO PROAÑO, D. J. Análisis comparativo de bases de datos de código abierto vs código cerrado (determinación de índices de comparación). Quito, Ecuador: 2006, 196 p. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.12.Articulo.Comparacion%20Bases%20de%20Datos%20Open%20y%20propietarias.pdf.

40. Oracle Berkeley DB 12c: [en línea], Disponible en:

http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

41. ALFONSO VALDÉS, J. G. P., ÁNGELA GLORIA; PIMENTEL GONZÁLEZ, LUIS ALBERTO. Desarrollo del módulo para la transmisión de datos de gran tamaño para el sistema Reko. [Artículo original]. Cuba: Serie Científica de la Universidad de las Ciencias Informáticas, 12 de diciembre del 2011,. vol. 4, 9 p. Disponible en: http://publicaciones.uci.cu/index.php/SC/article/view/490/486.

42. GARCÍA LIRA, K. G. D., AILEC Y TEJERA HERNANDEZ , DAYANA CARIDAD. Conferencia #1: Continuación de la Disciplina Análisis y Diseño. En Ingenieria de software II. Departamento de ISW, Universidad de las Ciencias Informáticas, Curso: 2010 - 2011. p. 56.

43. Suarez, Eivys Hernández. Administración de las Configuraciones y Definiciones de Seguridad del SIGEP. 2008.

44. Tema II. Fase de Construcción. En Conferencia 6. Disciplina de Implementación. Cuba, Universidad de las Ciencias Informáticas. 2011. p. 45.

45. Normas ISO25010, [en linea]. Disponible en: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35733

46. JACOBSON, I. B. G. R., JAMES El Proceso Unificado de Desarrollo de Software En Otero, A. (editor). Madrid: Pearson Eduaction S.A, 2000, vol. 1, p. 49.

47. SCHMULLER, J. Aprendiendo UML en 24 horas. Editado por: Educación, P. 2000, Disponible en: http://eva.uci.cu/mod/resource/view.php?id=9400&subdir=/Otros_materiales. ISBN 968-444-463-X.

48. MSDN. Desarrollo de software basado en componentes. 2013, Disponible en: http://msdn.microsoft.com/es-es/library/bb972268.aspx.

49. MSDN. Replicación de SQL Server. 2013, Disponible en:

http://msdn.microsoft.com/es-es/library/ms151198.aspx.

50**.** GARCÍA LIRA, K. G. D., AILEC; TEJERA HERNÁNDEZ, DAYANA CARIDAD. Arquitectura y Patrones de Diseño. En Tema I. Culminación de la Fase de Elaboración. Universidad de las Ciencias Informáticas, La Habana.

51. SOMMERVILLE, I. Requerimientos*.* En *Ingeniería del Software.* Traducido por: Alfonso Galipienso, M. I. 7 ed. Madrid: Pearson Education S.A, 2005, p. 712.

52. Patrón de Diseño DAO. 14 de semptiembre del 2011. [en línea]. Disponible en: http://j2ee.ibsi.cl/desarrollo/java-j2ee/patron-de-diseno-dao/.

53. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Editado por: Roig Vázquez, P. E. 1 ed. México: PRENTICE HALL, 1999. ISBN 970-17-0261-1.

54. Patrones del “Gang of Four”. Madrid: Facultad de informática - Universidad Politécnica de Madrid, Unidad Docente de Ingeniería del Software.

55. Joskowicz, Jose. “Reglas y Prácticas en eXtreme Programming” PearsonEducation, 2008.

56. CUEVA LOVELLE, J. M. Calidad de software. 1999, 12 p. Disponible en: http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

57. JURISTO, N., MORENO, ANA M., VEGAS, SIRA Técnicas de evaluación de software. 2005, Disponible en:

http://eva.uci.cu/file.php/257/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_3/Comun/Tecnicas_de_evaluacion_de_software_Jurisco-Moreno.pdf

ANEXOS

Anexo 1 Descripción atributo Fiabilidad de los RNF

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Tolerancia a fallos y Recuperabilidad.
Objetivo	Capacidad del producto para operar según lo previsto en presencia de fallos de hardware o software. Capacidad del producto para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallos.
Origen	Proveedor de requisitos.
Artefacto	El sistema y el código fuente.
Entorno	El sistema desplegado donde existen fallos en la red y eléctricos.
Estímulo **<<1>>. <<a>> <Recuperación ante un fallo de conexión al servidor de mensajería >**	**Respuesta: Flujo de eventos (Escenarios)**
Fallos de red.	1. El sistema se encuentra capturando y enviando acciones de réplica hacia el destino. 2. El sistema intenta enviar los datos, pero detecta que está desconectado el servidor de mensajería, detiene el proceso de envío de fichero hacia el nodo destino, hasta que se restablezca conexión con servidor de mensajería. La información del fichero a que se

	está replicando se encuentra persistida en una base de datos local. 3. El sistema se conecta al servidor de mensajería y reestablece el proceso de envío de fichero hacia el nodo emisor, restableciendo la conexión con el servidor de mensajería. 4. El sistema concluye el proceso antes afectado satisfactoriamente.
<<1>>. <<b>> <Recuperación ante un fallo del fluido eléctrico >	
Medida de respuesta	
Navegar en el sistema en presencia de fallo de la red.	

Anexo 2 Descripción atributo Fiabilidad de los RNF

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Madurez.
Objetivo	Capacidad del producto para satisfacer las necesidades de fiabilidad en condiciones normales.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado y funcionando en periodos de tiempos determinados.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>> <El sistema desplegado en los escenarios de réplica especificados por el cliente>	

El sistema funciona correctamente bajo condiciones y periodos de tiempos determinados por el cliente.	NA
Medida de respuesta	
Desplegar el sistema.	

Anexo 3 Descripción atributo Fiabilidad de los RNF

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Disponibilidad.
Objetivo	Capacidad del producto de estar operativo y accesible para su uso cuando se requiere.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado, funcionando las 24 horas del día.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>> <El sistema desplegado en los escenarios de réplica especificados por el cliente las 24 horas del día >	
El sistema funcionando en modo *background* como parte de un proceso demonio del sistema operativo.	NA
Medida de respuesta	
Desplegar el sistema.	

Anexo 4 Descripción atributo Mantenibilidad de los RNF

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Analizabilidad.

Objetivo	Facilidad con la que se puede evaluar el impacto de un determinando cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema.
Estímulo **<<1>>. <<a>>< Impacto de un determinado cambio sobre el resto del sistema>**	**Respuesta: Flujo de eventos (Escenarios)**
Se evalúa el impacto de un cambio en el sistema.	1. El arquitecto de software identifica los paquetes y clases implicados en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio. 2. Se evalúa el impacto partiendo de los resultados que arrojó el análisis anterior con respecto a la arquitectura del sistema.
<<1>>. <<b>>< Diagnosticar las deficiencias o causas de fallos en el sistema>	**Respuesta: Flujo de eventos (Escenarios)**
Se diagnostica las deficiencias o causas de fallos en el sistema.	1. El arquitecto de software identifica las posibles deficiencias o causas de fallos que se pueden originar en el sistema. 2. Se diagnostican las deficiencias o causas de fallos partiendo de los resultados que arrojó el análisis anterior.
<<1>>. <<c>>< Identificar las partes a	**Respuesta: Flujo de eventos**

modificar>	**(Escenarios)**
Se identifica las partes a modificar en el sistema.	1. El arquitecto de software identifica los paquetes y clases implicados en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio.
Medida de respuesta	
Analizar el cambio en el sistema.	

Anexo 5 Descripción atributo Mantenibilidad de los RNF

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Modificabilidad.
Objetivo	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>>< Capacidad de modificación del sistema>	
La arquitectura del sistema está diseñada para brindar facilidades a la hora de introducir modificaciones en el sistema. Esto permite que se puedan introducir cambios o modificaciones, que no afecten el correcto desempeño del resto de la solución.	NA
Medida de respuesta	

Introducir una modificación al sistema.

Anexo 6 Descripción atributo Mantenibilidad de los RNF

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Capacidad para ser probado.
Objetivo	Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.
Origen	Arquitecto de software y el analista
Artefacto	Diseños de casos de pruebas.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>>< Facilidad con la que se pueden establecer criterios de prueba para el sistema>	
Desde la concepción inicial del sistema se definen y delimitan las funciones según los requisitos funcionales y no funcionales a implementar especificando los argumentos o variables de entrada y salida del sistema, elementos que favorecen el proceso de identificación y elaboración de los distintos escenarios de prueba.	NA
Medida de respuesta	
Escenario de prueba del sistema.	

Anexo 7 Descripción atributo Portabilidad de los RNF

Atributo de Calidad	Portabilidad.
Sub-atributos/Sub-característica	Adaptabilidad.
Objetivo	Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Capacidad del sistema de adaptarse de forma efectiva a diferentes entornos>**	**Respuesta: Flujo de eventos (Escenarios)**
El sistema está diseñado con tecnologías que permiten que este se adapte a cualquier sistema operativo. Para llevar a cabo el proceso de réplica de fichero se hace necesario que todos los nodos tengan instalado el mismo sistema operativo. El *script* de inicio de la aplicación permite personalizar los recursos de RAM de uso de la aplicación. El fichero de configuración de las propiedades del sistema permite configurar los parámetros del sistema según las prestaciones que posea la estación de trabajo donde se encuentre instalado.	NA
Medida de respuesta	
El ambiente de despliegue del sistema.	

Anexo 8 Descripción atributo Portabilidad de los RNF

Atributo de Calidad	Portabilidad.
Sub-atributos/Sub-característica	Reemplazabilidad.
Objetivo	Grado en que un producto puede sustituir a otro producto de software especificado para el mismo propósito en el mismo entorno.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Facilidad con la que el sistema puede sustituir otros sistemas especificados para el mismo propósito>**	**Respuesta: Flujo de eventos (Escenarios)**
El sistema cuenta con las principales características definidas para los sistemas de réplica de datos en base de datos relacionales distribuidas.	NA
Medida de respuesta	
Analizar las características del sistema.	

Anexo 9 Descripción atributo Adecuación funcional de los RNF

Atributo de Calidad	Adecuación funcional.
Sub-atributos/Sub-característica	Integridad funcional.
Objetivo	Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificados.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Grado en el que el sistema**	**Respuesta: Flujo de eventos (Escenarios)**

cubre todas las tareas y objetivos especificados>	
El desarrollo del sistema esta guiado por las necesidades expresadas por parte de los proveedores de requisitos, dándole total cumplimiento a sus especificaciones declaradas e implícitas.	NA
Medida de respuesta	
Analizar las funcionalidades del sistema.	

Anexo 10 Descripción atributo Seguridad de los RNF

Atributo de Calidad	Seguridad.
Sub-atributos/Sub-característica	No repudio.
Objetivo	Grado en que las acciones o eventos pueden ser probados a haber tenido lugar, por lo que los eventos o acciones no pueden ser repudiados más tarde.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Sistema de eventos o trazas.>**	**Respuesta: Flujo de eventos (Escenarios)**
El sistema una vez iniciado crea un conjunto de trazas a nivel de ficheros en la raíz de la solución en una carpeta nombrada log donde se almacenan tres archivos : Audit.log: contiene todas las trazas relacionadas con las respuestas de la	NA

aplicación al cliente. Error.log: contiene los errores ocurridos en el sistema System.log: contiene todas las trazas relacionadas con las acciones ejecutadas por el sistema. Este sistema de trazas garantiza el no repudio sobre las acciones realizadas en el sistema.	
Medida de respuesta	
Navegar en el sistema.	

Anexo 11 Descripción atributo Eficiencia en el rendimiento de los RNF

Atributo de Calidad	Eficiencia en el rendimiento.
Sub-atributos/Sub-característica	Comportamiento en el tiempo.
Objetivo	Grado en que los tiempos de respuesta, procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Para escenario 1>**	**Respuesta: Flujo de eventos (Escenarios)**
Tiempo de respuesta.	1. El sistema funcionando correctamente.
Estímulo **<<1>>. <<a>>< Para escenario 2>**	**Respuesta: Flujo de eventos (Escenarios)**
Tiempo de respuesta.	1. El sistema funcionando correctamente.

Medida de respuesta
Navegar en el sistema.

Anexo 12 Descripción atributo Eficiencia en el rendimiento de los RNF

Atributo de Calidad	Eficiencia en el rendimiento.
Sub-atributos/Sub-característica	Utilización de recursos.
Objetivo	Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Características de Hardware >**	**Respuesta: Flujo de eventos (Escenarios)**
Servidor de aplicación • Cantidad: 1 servidores. • CPU: 4x2.33GHz. • RAM: 4Gb. • HDD: 200 Gb RAID 5. • LAN: 2 x NIC (10/100MBbit). • Fuentes de Alimentación: 2x500W.	1. El sistema funcionando correctamente.
Estímulo **< Características de Software >**	**Respuesta: Flujo de eventos (Escenarios)**
• Tener disponible al menos 2 puertos para el uso de la aplicación. • Instalador de la aplicación de REKO. • Instalador de la aplicación ActiveMQ. • Un navegador web (Firefox).	1. El sistema funcionando correctamente.
Medida de respuesta	
Navegar en el sistema.	

Anexo 13 Descripción de la clase PackageSendDaoImpl

Descripción de la clase PackageSendDaoImpl	
Nombre: PackageSendDaoImpl.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
database	private
configEnvirmoment	
enviromentFactory	
configBinding	private
Responsabilidades	
Nombre:	persist(list: List<PackageSend >)
Descripción:	Salva localmente a la agrupación en la BD embebida Berkerley.
Nombre:	removeId (id: String)
Descripción:	Dado un identificador elimina de la BD la lista de tipo PackageSend que corresponde con este.
Nombre:	getFirst(): List<PackageSend>.
Descripción:	Devuelve la primera lista de PackageSend que se tiene almacenado.

Anexo 14 Descripción de la clase PackageSendReceiverDaoImpl

Descripción de la clase PackageSendReceiverDaoImpl	
Nombre: PackageSendReceiverDaoImpl.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
database	private
configEnvirmoment	
enviromentFactory	
configBinding	private
Responsabilidades	
Nombre:	persist(list: List<PackageSend >)
Descripción:	Salva localmente a la agrupación en la BD embebida Berkerley.
Nombre:	removeId (id: String)

Descripción:	Dado un identificador elimina de la BD la lista de tipo PackageSend que corresponde con este.
Nombre:	getAll(): List<PackageSend>.
Descripción:	Devuelve la lista de PackageSend que se tiene almacenado.

Anexo 15 Descripción de la clase ControlNumberFileBuildDao

Descripción de la clase ControlNumberFileBuildDao	
Nombre: ControlNumberFileBuildDaoImp	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
database	private
configEnvirmoment	
enviromentFactory	
configBinding	private
Responsabilidades	
Nombre:	persist(value: String)
Descripción:	Salva localmente a la agrupación en la BD embebida Berkerley.
Nombre:	removeId (id: String)
Descripción:	Dado un identificador elimina de la BD la lista de tipo String que corresponde con este.
Nombre:	getAll(): List<String>.
Descripción:	Devuelve la lista de String que se tiene almacenado.

Anexo 16 Descripción de la clase NotifyManager

Descripción de la clase NotifyManager	
Nombre: NotifyManager	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
logger	protected
senderManager	private
deliverySender	private
receiverManager	private
Responsabilidades:	
Nombre:	sendNotification(notification : cu.uci.dbreplication.notify.Notification)
Descripción:	Envía una notificación.
Nombre:	onFileNotification(notification : cu.uci.dbreplication.notify.FileNotification)
Descripción:	Analiza las notificaciones recibidas de tipo FileNotification.
Nombre:	setSenderManager(senderManager : cu.uci.dbreplication.sender.SenderManager)
Descripción:	Permite modificar la instancia de la clase SenderManager.
Nombre:	setReceiverManager(receiverManager : cu.uci.dbreplication.receiver.ReceiverManager)
Descripción:	Permite modificar la instancia de la clase ReceiverManager.
Nombre:	afterPropertiesSet()
Descripción:	Método a implementar de la interfaz InitializingBean.

Anexo 17 Descripción de la clase FileNotification

Descripción de la clase FileNotification	
Nombre: FileNotification.	
Tipo de clase: Auxiliar.	
Atributo:	**Tipo:**
status	private
Target	private
toTarget	private

pathDeleteFile	private
FileCompleted	Public static
FileReceived	Public static
FileReceivedIncorrect	Public static
FileDelete	Public static
FileBuild	Public static
receivedPackageKey	private
Responsabilidades	
Nombre:	getSatus()
Descripción:	Evento ejecutado.
Nombre:	setStatus(target : String)
Descripción:	Permite modificar evento.
Nombre:	getTarget()
Descripción:	Obtiene nodo emisor de la notificación.
Nombre:	setTarget (transactionId : String)
Descripción:	Permite modificar el nodo emisor.
Nombre:	getToTarget()
Descripción:	Obtiene nodo destino.
Nombre:	setToTarget(toTarget: String)
Descripción:	Modifica el nodo destino.
Nombre	getReceivedPackageKey()
Descripción:	Obtiene el número del próximo paquete a enviar a un nodo destino.
Nombre:	set receivedPackageKey(int key)
Descripción:	Modifica el número del próximo paquete a enviar.

Anexo 18 Descripción de la clase JmsSenderFile

Descripción de la clase JmsSenderFile	
Nombre: JmsSenderFile.	
Tipo de clase: Auxiliar	
Atributo:	**Tipo:**

JmsTemplate	private
jmsConnectionWrapperUtil	private
Responsabilidades:	
Nombre:	sendFile (send : PackageSend, destino: String)
Descripción:	Envía mensaje mediante el servidor de mensajería a un destino dado.
Nombre:	auxSendFile (send : PackageSend, destino: String)
Descripción:	Envía mensaje mediante el servidor de mensajería a un destino dado.
Nombre:	afterPropertiesSet()
Descripción:	Método a implementar de la interfaz InitializingBean
Nombre	isConnected()
Descripción:	Devuelve si aplicación está conectada al servidor de mensajería.

Anexo 19 Descripción de la clase JmsReceiverFile

Descripción de la clase JmsReceiverFile	
Nombre: JmsReceiverFile.	
Tipo de clase: Auxiliar	
Atributo:	
receiverManager	
Responsabilidades:	
Nombre:	onFileArrive (package: PackageSend)
Descripción:	Analiza paquete recibido de tipo PackageSend.
Nombre:	onMessage (message : Message)
Descripción:	Recibe mensajes enviados.

Anexo 20 Descripción de la clase GeneralConfig

Descripción de la clase GeneralConfig	
Nombre: GeneralConfig.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
FTP_USE	protected

LOCALHOST_BASE_PATH	protected
DELETE_FILE_LOCAL	protected
FTP_MIN_SIZE	protected
BK_FILE_PATH	protected
BK_FILE_RECEIVED_PATH	protected
BK_BUILD_FILE_PATH	protected
Responsabilidades	
Nombre:	isFtpUse() boolean
Descripción:	Retorna si se hará uso de la réplica de ficheros.
Nombre:	isDeleteFileLocal() : boolean
Descripción:	Retorna si se desea eliminar fichero replicado directorio de ficheros si la acción ejecutada es eliminación.
Nombre:	getBkFilePath(): String
Descripción:	Retorna dirección de la BD embebida donde almacenará el fichero segmentado.
Nombre:	getBkFileReceivedPath(): String
Descripción:	Retorna dirección BD embebida donde se almacena paquetes recibidos.
	getBkBuildFilePath(): String
Descripción:	Retorna dirección de la BD embebida donde se lleva el control de los ficheros reconstruidos.
Nombre:	getBasePath(): String
Descrpción:	Retorna dirección del directorio de ficheros
Nombre:	getFtpMinSize(): int
Descripción:	Devuelve el tamaño por el cual se segmentará el fichero.

Anexo 21 Descripción de la clase GeneralConfigManager

Descripción de la clase GeneralConfigManager	
Nombre: GeneralConfigManager.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
Responsabilidades	
Nombre:	persist(generalConfigEntity: GeneralConfigEntity): void
Descripción:	Escribe datos en archivo replication.property.
Nombre:	getNodeId(): String
Descripción:	Obtiene el identificador del nodo.

Anexo 21 Descripción de la clase GeneralConfigEntity

Descripción de la clase GeneralConfigEntity	
Nombre: GeneralConfigEntity.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
ftpUse	Private
localHostBasePath	private
deleteFileLocal	private
ftpMinSize	private
BkSendFilePath	private
BkReceiverFilePath	private
Responsabilidades	
Nombre:	isFtpUse() boolean
Descripción:	Retorna si se hará uso de la réplica de ficheros.
Nombre:	isDeleteFileLocal() : boolean

Descripción:	Retorna si se desea eliminar fichero replicado directorio de ficheros si la acción ejecutada es eliminación.
Nombre:	getBkFilePath(): String
Descripción:	Retorna dirección de la BD embebida donde almacenará el fichero segmentado.
Nombre:	getBkFileReceivedPath(): String
Descripción:	Retorna dirección BD embebida donde se almacena paquetes recibidos.
Nombre:	getBasePath(): String
Descrpción:	Retorna dirección del directorio de ficheros
Nombre:	getFtpMinSize(): int
Descripción:	Devuelve el tamaño por el cual se segmentará el fichero.

Anexo 22 Descripción de la clase AplicatorTriggerDaoImpl

Descripción de la clase AplicatorTriggerDaoImpl	
Nombre: AplicatorTriggerDaoImpl.	
Tipo de clase: Auxiliar.	
Atributo:	**Tipo:**
Responsabilidades:	
Nombre:	execute(group: ReplicableGroup)
Descripción:	Aplica los cambios recibidos en la BD local.

Anexo 23 Descripción de la clase PackageSend

Descripción de la clase PackageSend

Nombre: PackageSend	
Tipo de clase: Auxiliar.	
Atributo:	**Tipo:**
package	Private
numberPackage	Private
packageTotal	Private
node	Private
fileName	Private
Responsabilidades:	
Nombre:	public PackageSend(byte[] package, int packageNumberr, int packageTotal, String node, String fileName)
Descripción:	Constructor de la clase.
Nombre:	getPackageTotal (): int
Descripción:	Devuelve el número de segmentos que contiene el fichero
Nombre:	getFileName(): String
Descripción:	Devuelve el nombre del fichero.
Nombre:	getPackageNumber(): int
Descripción:	Devuelve el número del paquete que se está analizando.
Nombre:	getPaackage(): byte[]
Descripción:	Devuelve segmento del fichero que se está replicando.
Nombre:	getNode(): String
Descripción:	Retorna nombre del nodo emisor.

Anexo 24 Descripción de la clase EnviromentFactory

Descripción de la clase EnviromentFactory	
Nombre: EnviromentFactory.	
Tipo de clase: Auxiliar.	
Atributo:	**Tipo:**
dbFileSenderEnviroment	Private
dbBuildFileEnviroment	Private
dbReceivedFileEnviroment	private

Responsabilidades:	
Nombre:	getDbReceivedFileEnviroment()
Descripción:	Devuelve objeto Enviroment para BD embebida dbBuildFileEnviroment.
Nombre:	getDbBuildFileEnviroment()
Descripción:	Devuelve objeto Enviroment para BD embebida dbBuildFileEnviroment.
Nombre:	getDbFileEnviroment()
Descripción:	Devuelve objeto Enviroment para BD embebida dbFileSenderEnviroment.

Anexo 25 Descripción de la clase FileUtil

Descripción de la clase FileUtil	
Nombre: FileUtil.	
Tipo de clase: Auxiliar.	
Atributo:	**Tipo:**
Path	Private
bufferSize	private
pathFile	Private
watchService	Private
nameFileInRoot	Private
Responsabilidades	
Nombre:	onFileSplitted(file: File): LinkedList<PackageSend>
Descripción:	Particiona un fichero
Nombre:	onBuildFile(LinkedList<PackageSend> packageReceived, String destino): File
Descripción:	Reconstruye un fichero.
Nombre:	LinkedList<PackageSend> onSplittedArray(int elementos, byte[] list, File fileName): LinkedList<PackageSend>

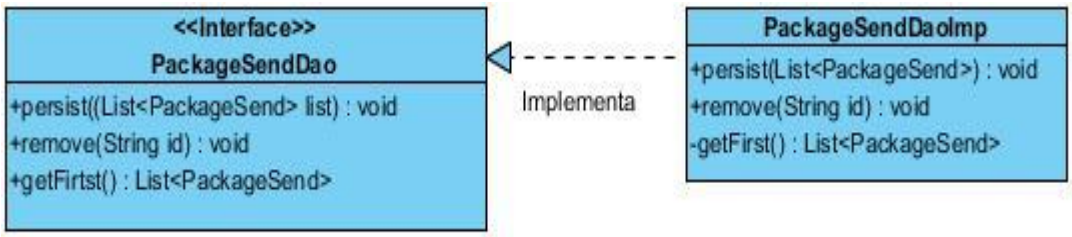
Descripción:	Particiona un arreglo de byte en segmentos.
Nombre:	scannedRootFile(): void
Descripción:	Escanea eventos ocurridos en un directorio de ficheros.
Nombre:	isFile(path: String): boolean
Descripción:	Devuelve si es un fichero.
Nombre:	isDirectory(path: String): boolean
Descripción:	Devuelve si es un directorio.
Nombre:	deleteFileMapControlAux(String file, int operation): void
Descripción:	Elimina evento del mapa donde se almacenan eventos ocurridos en el directorio de ficheros.
Nombre:	deletefileMapControl(ReplicableGroup group): void
Descripción:	Elimina evento del mapa donde se almacenan eventos ocurridos en el directorio de ficheros.

Anexo 26 Descripción de la clase SenderManager

Descripción de la clase SenderManager	
Nombre: SenderManager.	
Tipo de clase: Controladora.	
Atributo:	**Tipo:**
berkeleyPackageSenderDao	private
berkeleyNumberFileDao	private
onDestinyNodebuildFile	private
jmsFileSender	private
mapDestiny	private
Send_Completed	Private
packageMap	private
Targets	private
Responsabilidades	
Contiene todos los métodos get y set para obtener y modificar el atributo especificado respectivamente.	
Nombre	sendFileSplitted(action: cu.uci.dbreplication.transportable.ReplicableAction,

	transactionId: String, actionIndex: Integer)
Descripción	Envía fichero indexado en acción replicable a todos los nodos destinos.
Nombre	addNextPackage (idPaqueteEnviar: Integer, destino: String)
Descripción	Adiciona al mapa de destino próximo paquete a enviar con el identificador del nodo destino.
Nombre	cleanBDReceivedAndMap(paquete: LinkedList<PackageSend>, mapRemove: Map<Integer, PackageSend>).
Descripción	Borra información almacenada en BD embebida, y la información mapeada de la BD embebida.
Nombre	sendPackage(destino: String, paquete: PackageSend, transactionId: String).
Descripción	Envia objeto de tipo PackageSend hacia un destino definido.
Nombre	waitTemp()
Descripción	Verifica que existan destinos esperando próximo paquete a enviar.
Nombre:	onSendPackage(destino: String).
Descripción:	Envía a un destino determinado, un objeto de tipo PackageSend.

Anexo 27 Patrón DAO



Anexo 28 Patrón bajo acoplamiento